VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE PRO POŘIZOVÁNÍ NOVÝCH ZÁBĚRŮ HISTORICKÝCH FOTOGRAFIÍ

WEB APP FOR CAPTURING NEW SHOTS OF HISTORICAL PHOTOGRAPHS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MARTIN SIKORA**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. MARTIN ČADÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Sikora Martin, Bc.**
Obor: Informační systémy
Téma: **Webová aplikace pro pořizování nových záběrů historických fotografií**
Web App for Capturing New Shots of Historical Photographs
Kategorie: Web

Pokyny:

1. Seznamte se s problematikou re-fotografie, jejímž cílem je zachytit nový snímek odpovídající existující (historické) fotografii.
2. Navrhněte a implementujte systém pro re-fotografii (server, webové rozhraní). Při návrhu systému definujte požadavky na funkčnost a jednotlivé prvky prototypujte a testujte.
3. Se systémem experimentujte, posuďte jeho vlastnosti, proveďte uživatelský experiment a diskutujte možnosti budoucího vývoje.
4. Dosažené výsledky prezentujte formou videa, plakátu, článku, apod.

Literatura:

- dodá vedoucí práce

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Čadík Martin, doc. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této diplomové práce je návrh a implementace webové aplikace zaměřené na správu refotografií. Analyzovat stávající řešení, vytvořit seznam požadovaných funcí a jednoduché grafické uživatelské rozhraní. Součástí je také návrh a vytvoření API pro komunikaci s mobilní aplikací. Mezi základní požadavky aplikace patří zanesení fotografií do mapy, kombinování různých fotografií v grafickém editoru s rozšířenými funkcemi pro automatické zarovnání obrazu.

Abstract

The aim of this diploma thesis is to design and implement a web application focused on rephotography management. Analyze existing solutions, create list of features and simple graphical user interface. It also includes a design of API structure to communicate with the mobile application. Essential application requirements include adding photos on a map and combining different photos in a photo editor with enhanced auto-alignment features.

Klíčová slova

refotografie, opakovaná fotografie, webová aplikace, informační systém, historická fotografie, zarovnání fotografií, editor, PHP, elasticsearch

Keywords

rephotography, repeat photography, web application, information system, historical photography, photo alignment, editor, PHP, elasticsearch

Citace

SIKORA, Martin. *Webová aplikace pro pořizování nových záběrů historických fotografií.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Martin Čadík, Ph.D.

Webová aplikace pro pořizování nových záběrů historických fotografií

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Martina Čadíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Sikora
23. května 2018

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Doc. Ing. Martinu Čadíkovi, Ph.D. za rady a nasměrování při studiu zpracování obrazu, celkové vedení práce a možnost průběžně konzultovat řešení.

Obsah

Kapitola 1

Úvod

Prostředí měst se v dnešní době rychle mění, dochází k rekonstrukci budov a na nevyužitých plochách se vytváří parky nebo jiné objekty. Pokud máme atmosféru místa zachycenou na fotografii, můžeme pomocí refotografie, pohlédnout na stejnou scénu s časovým odstupem. Je však potřeba mít k dispozici historickou fotografii, abychom přesně věděli, z jaké pozice nový snímek vyfotografovat. Případně potřebujeme nástroj, obsahující funkce úpravy a transformací fotografií, pro lepší zarovnání snímků.

Podstatou této práce je navrhnout a implementovat aplikaci, umožňující uživatelům prohlížet fotografie zaznačené na mapě a nahrávat k nim nové verze. Kombinací historického snímku a refotografie vytvářet zcela nové obrázky, prolínající více období. K tomuto účelu je navržen editor s tranformačními funkcemi, pro ruční úpravu fotografií, automatickým zarovnáním a dalšími pomocnými nástroji.

Při nahrávání fotografií, systém zpracovává doplňující metadata, která následně využívá k různým účelům, např. získání času pořízení nebo souřadnic, pro automatické umístění na mapu. Pokud fotografie nebude obsahovat žádná nebo chybná metadata, je k dipozici dodatečná úprava.

Před samotným vývojem aplikace došlo k analýze existujících řešení (kapitola 4), kde výstupem bylo vypracovámí specifikace, tedy vypsání požadovaných funkcí aplikace a návrhy na vylepšení nedostatků existujících řešení.

Práce je koncipována jako webová aplikace z důvodu kompatibility s co nejvíce zařízeními. Nezaměřuje se na samotné pořizovaní refotografie, ale dodatečné zpracování a publikování. Na podporu zachycení refotografie, je možné vytvořit mobilní aplikaci a napojit ji na vystavené API, za účelem integrace služeb webového rozhraní do aplikace třetí strany, popsané v sekci 6.8.

Implementaci předcházel návrh řešení (kapitola 5), kde jsou popsány použité technologie a tvorba prototypu aplikace, na kterém proběho uživatelské testování, pro odhalení nedostatků grafického rozhraní.

Aplikace je dělena do třech částí. Uživatelské rozhraní, v kapitole 6, určené koncovému uživateli. Administrace, v sekci 6.7, sloužící na validaci vstupních dat a vyhodnocení používání díky statistikám a API umožňující otevřenou práci s daty portálu (sekce 6.8).

Kapitola 7 obsahuje serverové požadavky pro běh, případně je připravena konfigurace pro virtualizované spuštění pomocí kontejnerové platformy Docker.

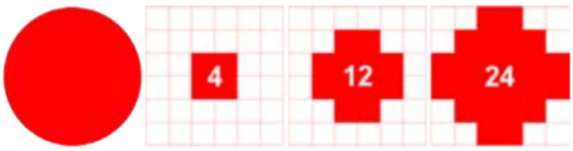
Součástí aplikace jsou automatizované testy, kontrolující správnost při úpravách. Došlo také k uživatelskému testování, ze kterého vyšla možná vylepšení popsána v kapitole 8.

Kapitola 2

Digitální fotografie

Jelikož se jedná o aplikaci zpracovávanou počítačem, je teorie zaměřená na digitální grafiku, její reprezentaci a možnosti uložení.

Digitální fotografie je reprezentována rastrovým, neboli bitmapovým obrazem. Ten je složen z dvojrozměrné mřížky bodů zvanými pixely. Velikost mřížky se nazývá rozlišení a určuje výslednou velikost obrazu. Větší rozlišení znamená více obrazových bodů a tedy detailnější zachycení scény. Příkladem je následující obrázek 2.1, znázorňující kruh při reprezentaci 4, 12 a 24 pixely. [11]



Obrázek 2.1: Reprezentace kruhu pomocí 4, 12 a 24 pixelů

Každý pixel nese informaci o barvě, většinou založené na kombinaci třech základních: červené (red), zelené (green) a modré (blue). Použití těchto barev odpovídá RGB modelu, využívající aditivní míchání barev, kdy se jednotlivé složky barev sčítají a vytvářejí světlo vyšší intenzity. Rozsah hodnot, které může složka nabývat, odpovídá barevné hloubce. Ta je definována číslem, značící počet bitů, rezervovaných pro uložení hodnoty. V následujícím výčtu je uveden počet barev pro n-bitové hlouby. [11]

- 1-bit: Monochromatické (2 barvy)
- 4-bity: Stupně šedi nebo barva (16 barev)
- 8-bitů: Stupně šedi nebo barva (256 barev)
- 16-bitů: High Colour (65 536 barev)
- 24-bitů: True Colour (16 777 216 barev)
- 32-bitů: True Colour (4 294 967 296 barev)

2.1 Zachycení

K pořízení digitální fotografie je potřeba fotoaparát s obrazovým snímačem. Tento snímač má za úkol zachytit intenzitu světla, působící na každou buňku senzoru. Při větším světle

dopadá na snímač více fotonů, které se následně spočítají a převedou na číslo. Posloupnost čísel je poté použita k rekonstrukci obrazu, určením barvy a intenzity každého pixelu. Jelikož senzor zachycuje pouze intenzitu světla a ne barvu, je potřeba filtr, který propustí pouze jednu barevnou složku záření. Poté pomocí interpolace spočítá fotoaparát výslednou barvu, kombinací hodnot získaných ze tří filtrů. Senzor obsahuje dvakrát více zelených filtrů, protože lidské oko je nejcitlivější na zelenou barvu. [11]

2.2 Komprese obrazu

Data získaná ze senzorou jsou paměťově náročná a tak před uložením dochází ke kompresi. Kompresní algoritmy jsou používány za účelem snížení počtu bitů potřebných k reprezentaci obrázku. Výstupem je tedy kompaktnější forma, která má menší paměťové nároky. Po použití komprese, nemusí být obrázek na první pohled ve viditelně horší kvalitě. Lidské oko má limity a data, která nerozpozná a v určitých případech je není potřeba uchovávat.

Kompresní techniky se řadí do dvou skupin. První technika generuje ze vstupu Γ komprimovanou reprezentaci Γ_c, obsahující méně bitů. Poté existuje rekonstrukční algoritmus, který transformuje komprimovaný vstup Γ_c na rekonstruovaný obraz Θ. [21]

Podle požadavků na rekonstrukci se kompresní schéma dělí na dvě skupiny - ztrátové a bezztrátové.

2.2.1 Ztrátová

Ztrátové schéma dovoluje, aby Θ nebyla totožná s Γ. Výsledný obraz je tedy aproximací původního. Komprimací může dojít například ke snížení množství barev, shlukováním sousedních pixelů nebo odstraněním nepotřebných dat.

Typicky jsou ztrátové formáty mnohem méně náročné na paměť, což z nich dělá ideální formát pro použití na webu. Nižší velikost má však za následek horší kvalitu výsledného obrazu, avšak algoritmy se snaží převést obrázek tak, aby byl okem nerozpoznatelný od originálu. [15]

2.2.2 Bezztrátová

U bezztrátového formátu se $\Theta = \Gamma$. Kompresí nedochází ke ztrátě dat a rekonstruovaný obraz je totožný s obrazem před aplikováním komprese. Formáty používající bezztrátovou kompresi, obsahují veškerá data původního obrázku. Nic z něj není odstraněno, z čehož plyne také název - *bezztrátové*. Soubor může být komprimován, ale na rozdíl od ztrátových formátu je i po komprimaci možné získat zpět originál. [15]

2.3 Souborové formáty

Po zachycení fotografie je potřeba mřížku pixelů a dodatečná data uložit. Formáty používají ztrátovou nebo bezztrátovou kompresi popsanou v sekci 2.2. V následujících podkapitolách jsou uvedeny případy, kdy použít daný typ pro uchování dat fotografie.

2.3.1 JPEG

JPEG sám není souborový formát, ale kompresní algoritmus pojmenovaný po vývojářké skupině *Joint Photographic Experts Group*. Ve spolupráci s *C-Cube Microsystems* vyvinuli

format *JPEG File Interchange Format*, zkráceně JFIF. Pro označení souboru využívající tento formát však zůstalo ozačení JPEG. Nejrozšířenějšími používanými příponami jsou *jpg* a *jpeg*.

Později byl zdokonalen formátem SPIFF, značícím *Still Picture Interchange File Format*. JFIF i SPIFF podporují pouze RGB modely s 24 bitovou barevnou hloubku a ve většině případů používají ztrátovou kompresi, avšak existuje i bezztrátová varianta. [9]

Díky kompresnímu algoritmu se velmi snižuje výsledná velikost souboru i kvalita. Je to nejrozšířenejší a nejvíce používaný formát obrázků používaný na webu. Díky své malé velikosti umožňuje rychlejší stažení a tím i načtení webové stránky. Je také vhodný jako příloha v emailu nebo pro tisk. Neumožňuje však průhlednost, kvůli chybějícímu alfa kanálu. [11]

2.3.2 JPEG 2000

JPEG 2000 je náhrada JPEG algoritmu vyvinuta skupinou *ISO JPEG group*. Definuje formát JP2 se stejným chováním jako JFIF a SPIFF. Umožňuje použít ztrátovou i bezztrátovou kompresi a používá diskrétní formu vlnkové transformace, pro získání větší komprese s menší degradací kvality obrazu. Výsledná komprese je o $20 - 30\%$ kvalitnější. [16]

Dovoluje použít maximálně 24-bitovou barevnou hloubku s využitím různých barevných modelů, ne pouze RGB. Na rozdíl od JPEG umožňuje pracovat s obrázky většími než 64000×64000 pixelů. Je mnohem méně používaný než původní JPEG. [15]

2.3.3 PNG

PNG, neboli *Portable Network Graphics*, používá bezztrátovou kompresi. Podporuje až 48-bit barevnou hloubku a pro každý pixel má navíc alfa kanál, který může nabývat hodnot $0 - 255$, definující průhlednost. Díky tomu je velmi rozšířen na webu např. pro zobrazení loga nebo různých grafických prvků, které nevyplňují celou plochu obrázku. Tento formát je optimalizován pro obrazovky [15]

2.4 Metadata

Při vytváření snímku ukládá fotoaparát dodatečné informace - metadata. Ty jsou vloženy přímo do hlavičky souboru s obrázkem a jsou definovány standardem EXIF (Exchangeable Image File format). Standard se stále vyvýjí a je spravován asociací JEITA (Japan Electronics and Information Technology Industries Association). Metadata mohou být také ručně přidána nebo upravena pomocí fotografických editorů.

Mezi ukládaná data patří např. datum a čas pořízení snímku, nastavení fotoaparátu (citlivost ISO, clona), označení fotoaparátu a GPS souřadnice místa. Mohou tak velmi ulehčit kategorizaci snímků nebo zpětné vyhodnocení chybného nastavení při fotografování.

Metadata nepodporuje JPEG 2000, ale formáty JPEG, TIFF, RAW a nově i PNG ano. [11, 17]

2.5 Refotografie

Refotografie nebo opakovaná fotografie (repeat photography) je proces zachycení scény, která již byla před nějakým časem zaznamenána. Důležité je stejné postavení fotografa a nastavení ohniskové vzdálenosti jako na prvnotním snímku. Není definován časový rozestup

mezi snímky, ale nová fotogragie by měla být zřetelně jiná. Díky refotografii můžeme mapovat změny městských částí, rekonstrukce objektů a celkově změnu prostředí daného místa v čase. [8]

Vytvoření naprosto stejné fotografie je obtížký úkol. Usnadněním mohou být například mobilní aplikace napomáhající zachycení tím, že starou fotografii promítnou do aktuálního zorného pole fotoaparátu a umožní fotografovi v reálném čase vidět, jestli jsou hrany zarovnány. Pro přesnější zarovnání je možno takto zachycené digitální fotografie dodatečně transformovat v editoru.

Obrázek 2.2: Ukázka použití refotografie.

Kapitola 3

Práce s obrazem

Při vytváření refotografií se většinou nepodaří získat fotografii přesně odpovídající originálu. Díky fotografickým editorům však můžeme obrázek upravit tak, aby např. hrany budov byly zarovnány. Každá fotografie je dvojrozměrná množina pixelů, kterou si můžeme představit jako 2D rovinu složenou z bodů.

2D bod je definován jako dvojrozměrný vektor $\mathbf{x} = (x, y) \in \mathbb{R}^2$. Maticová reprezentace je znázorněna v rovnici (3.1)

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.1}$$

Bod může být také reprezentován pomocí Homogenních souřadnic $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathcal{P}^2$, kde jsou za ekvivalentní považovány vektory, lišící se pouze ve váze. $\mathcal{P}^2 = \mathbb{R}^3 - (0, 0, 0)$ se nazývá 2D projekční prostor.

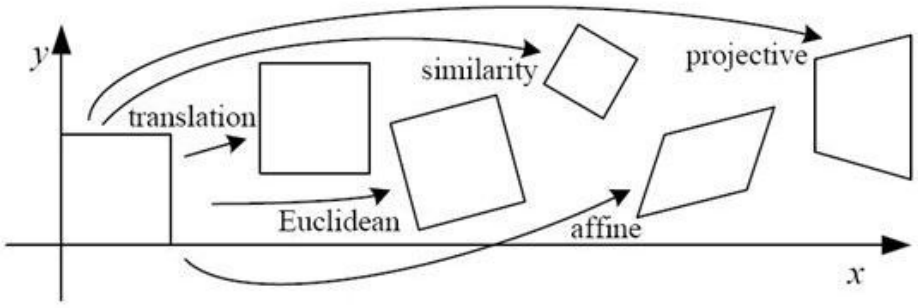
Homogenní vektor $\tilde{\mathbf{x}}$ může být převeden zpět do nehomogenního vektoru $\mathbf{x}$ dělením podle poslení složky $\tilde{w}$, tj.

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{x} \tag{3.2}$$

kde $\bar{x} = (x, y, 1)$ je *rozšířený vektor*. Homogenní body, jejichž poslední složka $\tilde{w} = 0$ se nazývají *body v nekonečnu* a nemají ekvivalent v nehomogenní reprezentaci. [23]

3.1 2D Transformace

Tato sekce popisuje 2D transformace bodů zobrazené na obrázku 3.1.



Obrázek 3.1: Ukázka 2D rovinných transformací. [23]

3.1.1 Posunutí

2D posunutí může být zapsáno jako $x' = x + t$ nebo

$$x' = \begin{bmatrix} I & t \end{bmatrix} \bar{x} \tag{3.3}$$

kde I je jednotková matice (2×2).

Přidáme-li na konec matice nový řádek $\begin{bmatrix} 0^T & 1 \end{bmatrix}$, dostaneme matici (3×3), se kterou je možné spojovat transformace pomocí maticového násobení. Výsledná transformace je pak zapsána na rovnici 3.4. [23]

$$\bar{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \bar{x} \tag{3.4}$$

3.1.2 Euklidovská transformace

Tato transformace je kombinací rotace a posunutí. Nazývá se euklidovskou, kvůli zachování euklidovké zvdálenosti po aplikaci. Může být zapsána jako $x' = Rx + t$ nebo

$$x' = \begin{bmatrix} R & t \end{bmatrix} \bar{x} \tag{3.5}$$

kde

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.6}$$

je ortonormální matice rotace s $RR^T = I$ a $|R| = 1$. [23]

3.1.3 Transformace podobnosti

Jedná se o euklidovskou transformaci se změnou měřítka definovaou jako $x' = sRx + t$, kde s je faktor měřítka. Maticový zápis je uveden v rovnici 3.7.

$$x' = \begin{bmatrix} sR & t \end{bmatrix}, \bar{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{x} \tag{3.7}$$

Není potřeba, aby $a^2 + b^2 = 1$. Transformace podobnosti zachovává velikosti úhlů mezi přímkami. [23]

3.1.4 Afinní transformace

Afinní transformace je kombinací posunutí, rotace, změny měřítka a zkosení. Zapisuje se jako $x' = A\bar{x}$, kde A je matice (2×3)

Samotné zkosení [12] podle osy x definujeme jako $x' = x + sh_x y$ nebo

$$\bar{x}' = \begin{bmatrix} 1 & 0 & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{x} \tag{3.8}$$

respektivě podle osy y jako $y' = y + sh_y x$ nebo

$$\bar{x}' = \begin{bmatrix} 1 & sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{x} \tag{3.9}$$

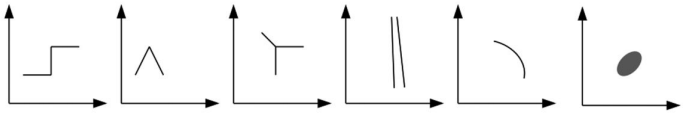
3.2 Automatická tranformace

Přestože má uživatel možnost pomocí grafického editoru zarovnat korespondující objekty na fotografiích, je možné tuto akci částečně nebo plně automatizovat. Důležitým aspektem je detekce klíčových bodu fotografie. Tyto body může uživatel zadat ručně, ale existují i algoritmy pro hledání bodů v obraze.

Pro automatické zarovnání fotografií je potřeba získat speciální body obrázku. Využitím algoritmů počítačového vidění, je možné tyto body z fotografie získat, následně porovnat mezi sebou body nalezené ve dvou různých fotografiích a určit shody, pomocí kterých vypočítáme transformační matici. Aplikací matice na fotografii získáme transformovaný obraz, který by měl být zarovnán stejně jako původní fotografie.

3.2.1 Klíčové body

Klíčovými body označujeme zajímavé body obrázku: rohy, hrany nebo kontury a velké vlastnosti plochy. Příklady jsou uvedeny na obrázku 3.2. Klíčové body jsou invariantní k rotaci a změně měřítka. Ikdyž jsou obrázky transformované, je možné najít korespondence. Body by měly být snadně vyhledatelné a umístěny na *dobrém* místě, pro jednoduchý výpočet deskriptoru, popsaného v kapitole 3.2.3. [14]



Obrázek 3.2: Typy klíčových bodů. [14]

3.2.2 Detekce klíčových bodů

Obecně používají detekční algoritmy minimální a maximální body, jako např. gradientní vrcholy, pro získání rohů a hran určující klíčové body. Použití rohů je častokrát upřednostněno před hranami nebo izolovanými maximi, z důvodu možného výpočtu úhlové orientace pro klíčový bod.

Body je možno počítat jak na barevném snímku tak v odstínech šedi. Mnoho algoritmů nejprve aplikuje Gaussovský filtr a poté operátor gradientu. Filtr se používá pro snížení šumu v obraze, který by byl jinak zvýrazněn. [14]

Pro detekci klíčových bodů existuje mnoho algoritmů počítačového vidění. Některé jsou volně dostupné k využití např. FAST, jiné jsou patentované a je potřeba zakoupit licenci, např. SIFT a SURF. Neexistuje však metoda použitelná pro všechny aplikace. [23]

FAST

FAST (Features from Accelerated Segment Test) spoléhá na spojenou sadu pixelů v kruhovém vzoru pro určení rohu. Velikost oblasti je obvykle 9 nebo 10 z možných 16. FAST je rychlý na výpočet s docela dobrou přesností, což z něj dělá ideálního kandidáta na použití v aplikacích se zpracováním v reálném čase.

Používá binární porovnání každého pixelu v kruhu se středovým pixelem a pomocí prahové hodnoty určí, zda je pixel menší nebo větší než středový. Výsledný deskriptor je uložen jako bitový vektor v pořadí od 0 do 15. [14]

3.2.3 Deskriptory

Po nalezení klíčových bodů, přichází na řadu *spojování*. Nejprve je potřeba určit, který klíčový bod odpovídá dané pozici v druhém obrázku. K tomuto určení je nutno získat deskriptor klíčového bodu, což je popis oblasti kolem něj. Jedná se o vektor. se zakódovanou informací o okolí, který je invariantní k transformacím. [14]

BRIEF

Motivací k vytvoření BRIEF descriptoru byl fakt, že SIFT používá 128 dimenzionální vektor, obsahující čísla s plovoucí desetinou čárkou pro popis deskriptoru, což při výpočtu tisíců bodů potřebuje značné množsví paměti a výpočetních prostředků. Všechny dimenze však nemusí být využity při porovnávání a je tedy možné použít kompresi a převod desetiných čísel na binární řetězec. Tyto řetězce používají k mapování Hammingovu vzdálenost, která má jednoduchý výpočet pomocí operace XOR a dalších menších výpočtů. Řetězce jsou však počítány z deskriptorů a problém s pamětí to neřeší. [14]

BRIEF však umožní získat přímo binární řetězce bez hledání deskriptorů. Vezme vyhlazený obraz a vybere množinu bodů n_d. Na této množině provede porovnání intenzit jednotlivých pixelů. Výsledek každého porovnání je binární hodnota, reprezentující zda byla intenzita větší nebo ne. Spojením výsledků dostáváme n_d-dimenzionální binární řetězec. [14]

3.2.4 Výpočet descriptoru

Tato sekce popisuje získání deskriptoru pomocí algoritmu ORB, který je spojením detektoru FAST a descriptoru BRIEF s modifikacemi pro zvýšení výkonnosti. Nejprve používá FAST pro nalezení klíčových bodů obrazu a poté aplikuje *Harris corner measure* [23] pro vyfiltrování pouze několika nejlepších. Problémem je, že FAST nepočítá orientaci, takže není invariantní vůči rotaci. Vylepšení algoritmu spočívá ve výpočtu vážené intenzity geometrického středu oblasti s rohem umístěným ve středu. Směr vektoru. od tohoto rohu do geometrického středu, určuje orientaci.

ORB je značně rychlejší než SURF a SIFT. Podle testování na skupině 24 obrázků s NTSC rozlišením, byla doba výpočtu: ORB 15.3ms, SIFT 217.3ms a SURF 5228.7ms. [14]

3.2.5 Nalezení korespondencí

Nejjednoduší cestou je porovnat každý klíčový bod v prvním obrázku s každým klíčovým bodem v druhém. Časová složitost tohoto porovnávání je však kvadratická a algoritmus tak není vhodný pro použití ve většině aplikací.

Lepším řešením je vytvořit indexovanou stukturu, např. multidimenzionální vyhledávací strom nebo hashovací tabulku, díky které dojde k jednoduššímu hledání klíčových bodů v oblasti kolem jiného. Stukturu je možno vytvořit jak pro každou fotografii zvlášť, což je výhodné při hledání konkrétního objektu, tak globálně pro databázi. [23]

Multidimenzionální hashování Rozděluje deskriptory na části o fixní velikosti, podle funkce aplikované na vektor deskriptoru. Při porovnávání je každý nový deskriptor vložen do určité části a potencionální kandidáti jsou vybírání ze sousedních částí. [23]

Kapitola 4

Existující řešení

Aktuálně neexistuje webová aplikace, zaměřující se na refotografie s vlastním editorem a dalším zpracováním fotografií. Jedná se buď o samostaný grafický editor (sekce 4.1) nebo pouze galerii refotografií (sekce 4.2).

Z obou těch kategorií bylo vybráno pár zástupců a analýzou zjištěno, jaké funkce uživatelé nejčastěji využívají s ohledem na odhalení nedostatků. Výstupem je seznam požadovaných funcí aplikace a návrh vylepšení slabin existujících řešení, popsaný v sekci 4.3.

4.1 Fotografické editory

Jsou to aplikace určené k editaci a tvorbu obrázků. Umožňují pouze transformaci fotek a aplikováním dalších funkcí vytvořit nový obrázek, který je možné uložit do zařízení.

4.1.1 Fotor

Fotor[1] se primárně zaměřuje na úpravu jedné fotografie, za účelem přidání do vlastní lokální knihovny. Obsahuje funkce pro rotaci, změnu velikosti a aplikování různých filtrů, ale neumožňuje fotografii deformovat tak, aby odpovídala jiné.

Uživatelské rozhraní je jednoduché, jednotlivé funkce jsou tříděny do kategorií v levém menu. Po vybrání dojde k zobrazení konkrétních funkcí. Tento panel spolu s horním menu a pravým výčtem fotografií zabírá hodně místa, což způsobuje malou pracovní plochu. Je tedy nutné při práci přibližovat obrázek. Při přiblížení se zobrazí zmenšenina s vyznačenou částí, kde se aktuálně nacházíme, která je použitelná, ovšem opět zmenší pracovní plochu.

Aplikace je nejspíše vyvýjena pimárně pouze pro webový prohlížeč Chrome, protože při návštěvě z jiného prohlížeče dojde k zobrazení zprávy: *Please use Chrome fo best experience with Fotor. You may encounter some problems with your current browser.*

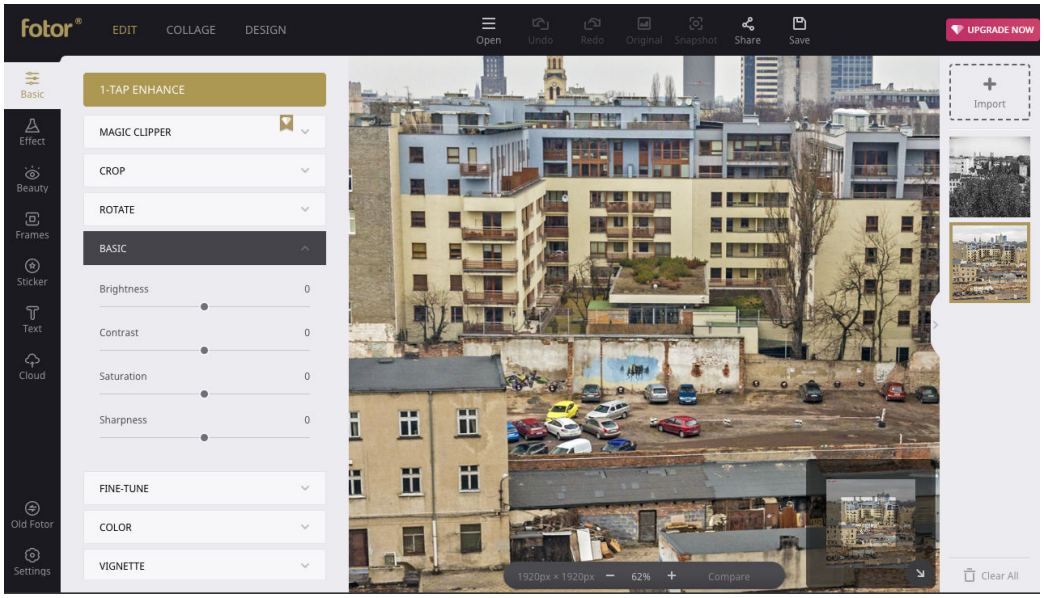
4.1.2 Photopea

Dalším editorem je Photopea[2]. Na první pohled vidíme, že je aplikace velmi inspirována programem Photoshop [4]. Oproti Fotor-u je mnohem propracovaňější, má více funkcí, od transformací, přes úpravu barev až po práci s vrstvami. Právě vrstvy jsou velkou výhodou této aplikace, jelikož umožňujé přidat více fotografií, z výběru vytvořit nové vrstvy a fotky poté prokládat.

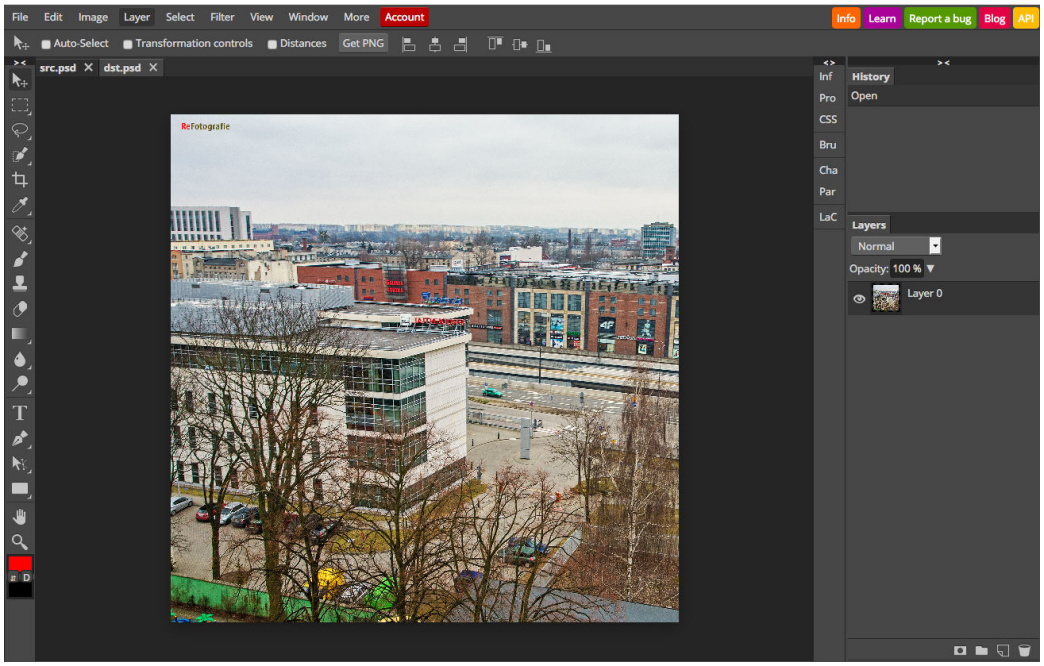
[1] Dostupné na http://fotor.com
[2] Dostupné na https://www.photopea.com

Aplikace však neumí automatické zarovnání fotek, připadně hledání podobností a uživatel si tak fotografie musí zarovnat sám. Což může být nemožné, protože z transformací jsou zde pouze funkce pro změnu velikost a zakřivení, nikoli však deformaci pomocí sítě bodů.

Uživatelské rozhraní je velmi intuitivní, jednotlivé funkce jsou dostupné pod výstižnými ikonami a každá akce vytvoří záznam v historii a je tedy možné aplikované změny vrátit.



Obrázek 4.1: Uživatelské rozhraní aplikace Fotor



Obrázek 4.2: Uživatelské rozhraní aplikace Photopea

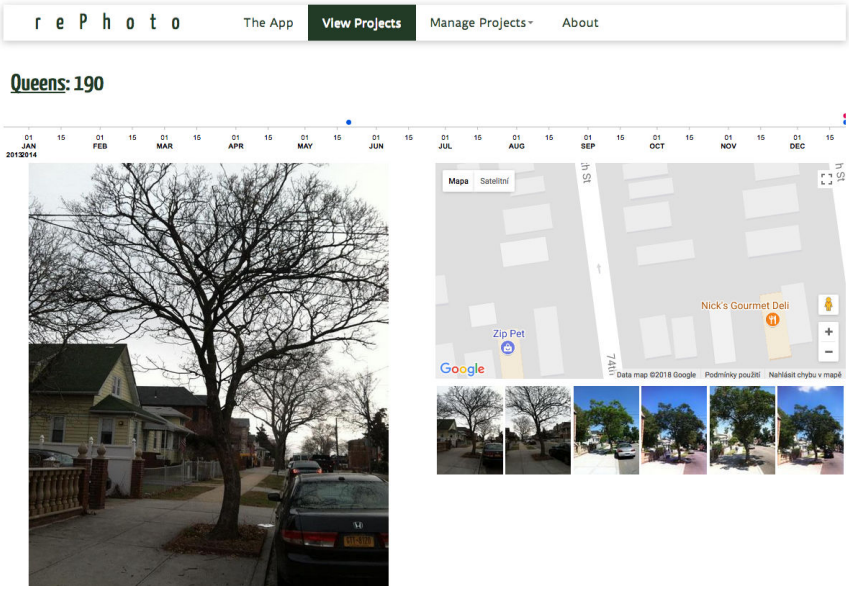
4.2 Galerie rephotografií

Webové systémy pro práci s refotografiemi předpokládají, že uživatel má vyfotografovanou přesnou fotografii nebo k úpravě došlo v jiném programu. Jedná se pouze o galerie, do kterých je možné přidat nové zachycení dané oblasti a prohlížet příspěvky jiných uživatelů.

4.2.1 Project rePhoto

Tento projekt[3] je především zaměřený na mobilní aplikaci, která zjednodušuje zachycení refotografie. Na webu jsou hlavním prvkem projekty, kde každý mapuje určitou oblast a obsahuje fotografie zaznačené na mapě. Není zde mapa světa, kde by byl přehled všech fotografií, ale pouze seznam projektů. Každé místo v projektu má detail se všemi fotografiemi a časovou osou, znázorňující datum pořízení fotografie.

Rozhraní je velmi jednoduché, ašak podle mého názoru ne příliš zdařilé. Neexistuje zde vyhledávač projektů. Na první pohled není zřejmé, jaká místa projekt obsahuje a některé prvky nepůsobí jako aktivní, uživateli tak může zůstat část webu skryta.



Obrázek 4.3: Uživatelské rozhraní aplikace Project rePhoto

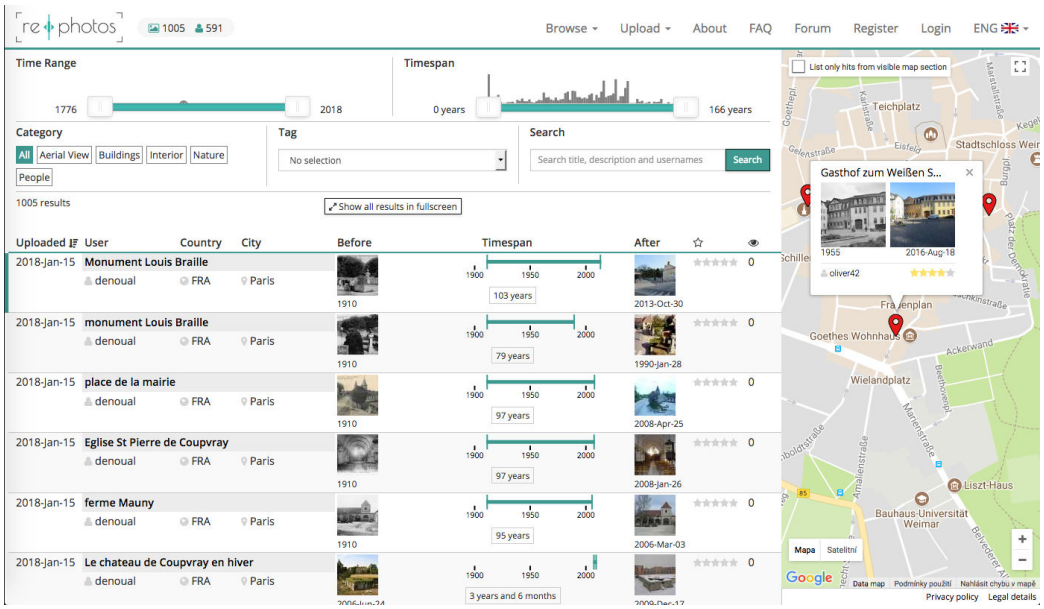
4.2.2 re.photos

Re.photos[4] je jediná aplikace, kde při náhrání refotografie dochází k automatickému zarovnání podle historické fotografie. Pokud nejsou správně zarovnány, uživatel může zvolit odpovídající body a spustit zarovnání znovu. Kombinovat lze vždy pouze dvě fotografie, pro vytvoření verze *před a po*. K jedné fotografii tedy nelze nahrát více než jedna nová a pro přidání další je zapotřebí vytvořit novou dvojici, *před a po*.

Fotografiím lze nastavit kategorii a značky pro lepší kategorizaci a prohlížení relevantních záznamů. Hlavní stránka obsahuje několik filtrů a následně seznam dvojic fotografií s informacemi. Po kliknutí na řádek dojde k otevření nové stránky - detailu. Zde je zobrazena

[3] Dostupný na http://projectrephoto.com/
[4] Dostupné na https://www.re.photos

levá polovina staré fotografie a pravá polovina nové. Vertikálním posuvníkem můžeme měnit poměr zobrazených fotografií. Toto je jediná aplikace použitelná pro zobrazení refotografií a nelze ji měnit.



Obrázek 4.4: Uživatelské rozhraní aplikace re.photos

4.3 Výstup analýzy

Prozkoumáním existujících řešení jsem vytvořil počáteční specifikaci aplikace. V systému budou figurovat dvě uživatelské role: uživatel a administrátor.

Uživatelská sekce systém bude obsahovat 4 hlavní části: mapa, editor, profil uživatele a informační stránky.

Úvodní stránka, tzv. landing page, bude popisovat funkce aplikace a ukázku vytvořené refotografie. Velké tlačítko *Procházet fotografie* povede na katalog fotografií s mapou.

Na mapě půjde vyhledávat podle adresy či místa a při změně okna mapy dojde k automatické změně fotografií v náhledu. Pro rychlou a snadnou práci s fotografiemi bude, po kliknutí na fotografii v mapě, zobrazen detail s doplňujícími informacemi a případně tlačítky pro další funkcionalitu. Všichni uživatelé uvidí tlačítko - přidání fotografie do editoru. Přihlášení uživatele zde bude mít navíc uložení fotky do oblíbených, tlačítko pro rychlé přidání nových fotografií a upozornění na nepublikované fotografie.

Po vybrání fotografií se uživateli zpřístupní stránka editoru. Zde bude možnost na fotografie aplikovat afinní transformace, měnit průhlednost, označovat části fotek a vytvářet z nich nové vrstvy. Také bude možné aplikovat automatické zarovnávání. Výsledný obrázek bude možné stáhnout do zařízení nebo uložit do systému, to však pouze pro přihlášené uživatele.

Profil uživatele seskupuje stránky pro přihlášení s možností zaregistrovat se přes email a nebo použití účtu ze sociální sítě Facebook. V samotném profilu budou zobrazeny oblíbené, uživatelem nahrané fotografie a výsledné obrazy z editoru.

Kapitola 5

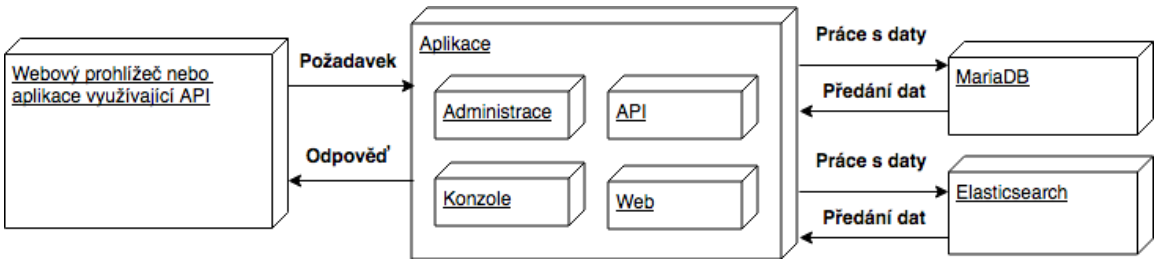
Návrh a použité technologie

V předchozí kapitole byly analyzovány programy s podobným zaměřením a výstupem je seznam funkcí, které by měl portál obsahovat. V následujících sekcích je navržena struktura systému a popsány konkrétní části aplikace. Obsahem je také ukázka a testované případy prototypu.

Dále je popsána architekturu systému, použité technologie a některé zajímavé implementované prvky.

5.1 Struktura systému

Aplikace je realizována jako webová, to znamená, že není závislá na operačním systému ani specifickém hardwaru. Motivací bylo vytvořit systém, dostupný pro co nejvíce nažízení s nejméně nároky. Hlavní část aplikace se nachází na serveru, oddělena od konečného zařízení, kde dochází k výpočtům, uchování a přípravě dat. Koncové zařízení je tak pouze zobrazovacím prvkem a nejsou na něj kladeny velké nároky.



Obrázek 5.1: Schéma částí systému a komunikace

5.1.1 Webová aplikace

Adresářová struktura aplikace vychází z použití Yii2 frameworku a je doplněna o vlastní adresáře pro testy, skripty a nahrané soubory. Význam adresářů a souborů je popsán v příloze B.

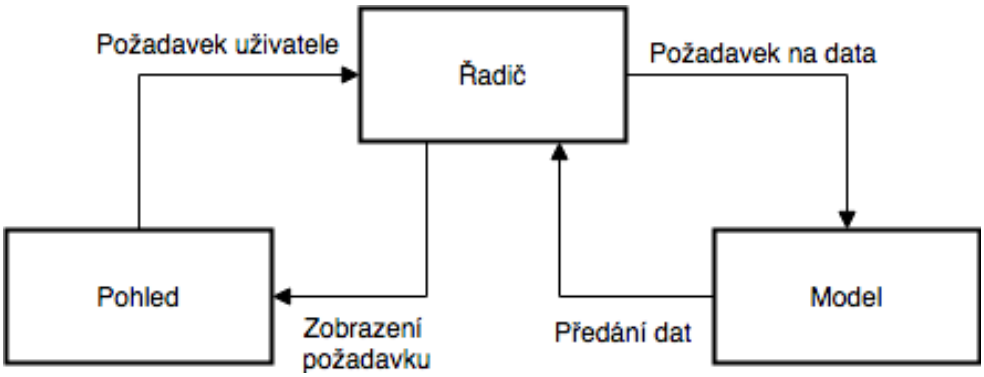
PHP

Většina serverové logiky informačního systému je naprogramována v jazyce PHP 7.1[1]. Jedná se o serverově orientovaný jazyk, kde jsou všechny výpočty prováděny na straně serveru, čím se nekladou zbytečně velké nároky na požadovaný výkon klientského zařízení. PHP je netypovaný jazyk, proměnné tedy nemají předem definovaný datovy typ a tak se v průběhu prohramu může měnit.

Framework Yii2

Aplikace je vystavěna na PHP frameworku Yii2[2]. Ten přináší ulehčenou práci např. při komunikaci s databází, zavedením modelů, což jsou objekty obsahující metody pro získání dat z konkrétní tabulky a automatickým nastavením atributů podle schéma databáze. [2]

Framework je založen na návrhovém vzoru Model-View-Controller (MVC) [2, 10]. Při dotazu uživatele na server přijde požadavek do řadiče (controller), kde dochází k analýze, vytvoření požadovaných modelů pro získání dat a řízení se jim předá. Modely následně komunikují s databází, a provádějí potřebné výpočty. Výsledek vrací zpět řadiči, který jej předá dál pohledu (view). Pohled obsahuje informaci o struktuře výstupu a doplní data na potřebná místa. Výsledek se přenáší zpět uživateli. Díky tomu je kód strukturován do částí a tím je zajištěna lepší přehlednost průběhu zpracování požadavku.



Obrázek 5.2: Diagram návrhového vzoru MVC [22]

Python

Skripty pro automatické získání klíčových bodů obrazu a automatické zarovnání jsou napsány v jazyce Python, protože obsahuje lepší provázání s knihovou OpenCV než PHP, pro které oficiální podpora neexistuje a uživatelské pokusy nejsou moc zdařilé nebo neobsahují funkce potřebné pro tuto aplikaci.

Pro běh aplikace je nutné mít na serveru nainstalován Python verze alespoň 2.7 a vytvořeno vyrtuální prostředí s nainstalovanou knihovnou Numpy[3] a OpenCV2[4].

[1] Dostupný na http://php.net
[2] Yii2 https://www.yiiframework.com/
[3] Numpy dokumentace https://docs.scipy.org/doc/numpy/dev/
[4] OpenCV dokumentace https://docs.opencv.org/2.4/modules/refman.html

5.1.2 Uživatelské zařízení

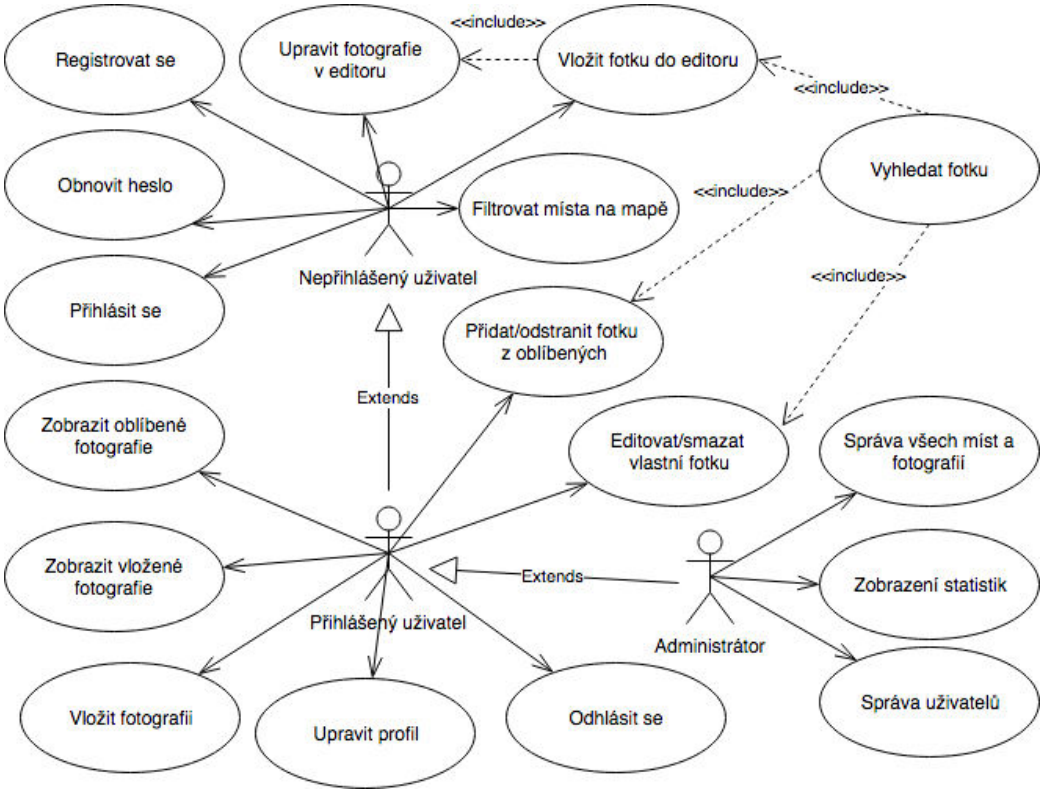
Protože se jedná o webovou aplikaci, jednotlivé a požadavky jsou zasílány z webového prohlížeče. Systém je vyvíjen s ohledem na nejrozšířenější prohlížeče a jsou použity takové konstrukce, aby stránka vypadala a fungovala stejně, bez ohledu na prohlížeč. Podporovanými prohlížeči s minimální verzí jsou:

- Chrome 66
- Safari 11
- Edge 17
- Firefox 60
- Opera 50

5.2 Uživatelé

Systém bude obsahovat dvě role a tři typy uživatelů. Uživatelská role pro běžnou práci s portálem a administrátorskou pro celkový přehled a správu. Třetím typem uživatele je nepřihlášený uživatel, který má omezenější práva oproti přihlášenému. Nemůže vytvářet vlastní fotografie a publikovat kombinované obrázky z editoru.

Povolené akce pro každý typ uživatele jsou znázorněny v diagramu případu užití [20] na obrázku 5.3.

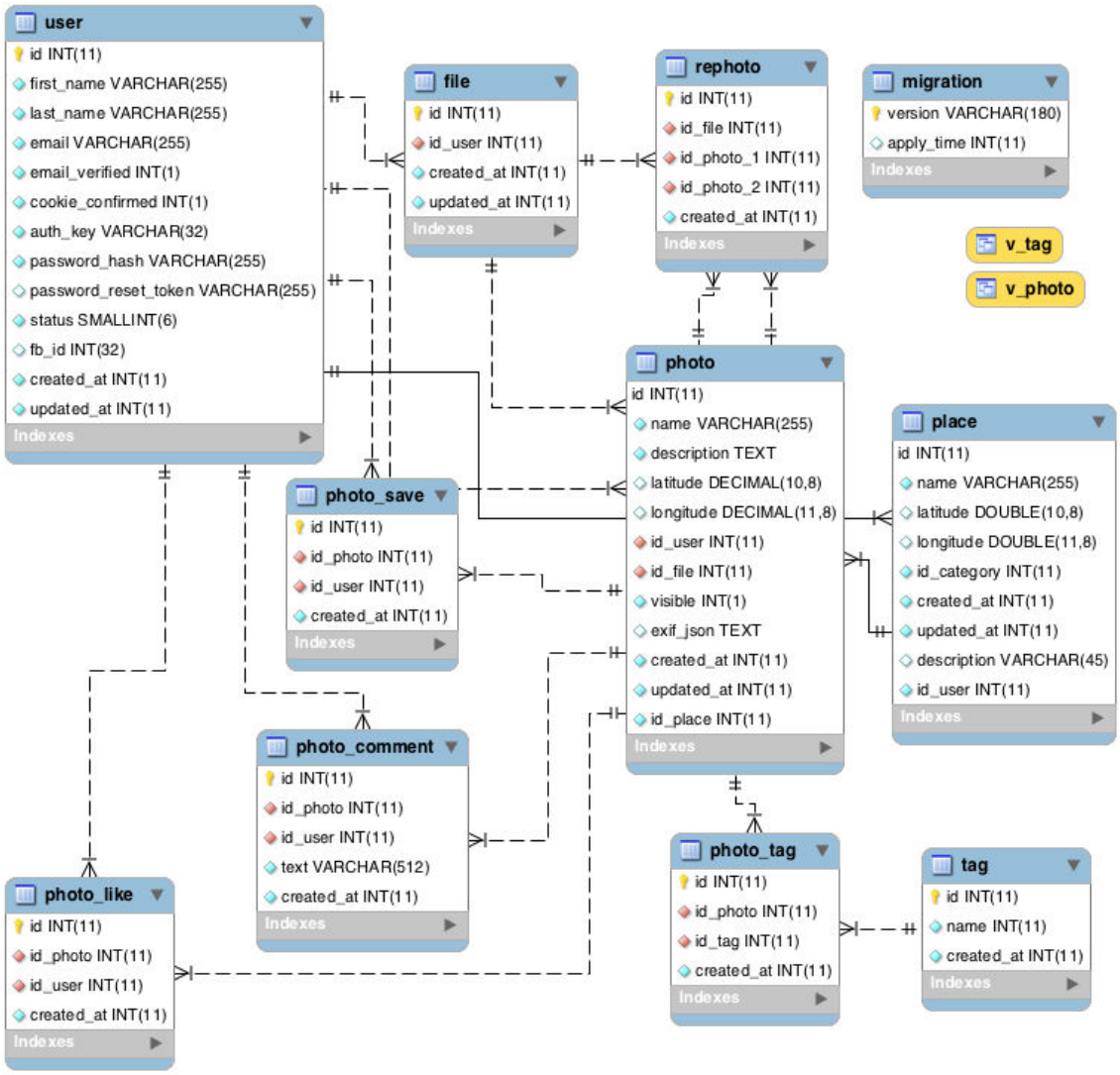


Obrázek 5.3: Diagram případů užití

5.3 Databáze

Každý informační systém pracuje s daty, která je potřeba uchovávat. Tím že jsou dynamicky přidávány, upravovány a filtrovány nemohou být staticky v kódu. Systém a struktura uložení by měla být navržena tak, aby přístup k nim trval krátkou dobu a data byla stálá.

Pro perzistentní uložení dat je k dispozici relační databáze. Schéma databáze je zobrazeno na obrázku 5.4. Tabulka *migration* slouží pro uchování aplikovaných databázových migrací. Kvůli zpětné kompabilitě a jednoduché aktualizace je každá změna ve struktuře databáze reprezentována migrací.



Obrázek 5.4: ER diagram databáze

Data uživatelů jsou uchována v tabulce *user*, fotografie ve *photo* a vazební tabulky *photo_like*, *photo_save* pro oblíbené, resp. uložené fotografie a *photo_comment* obsahující komentáře.

K dispozici jsou pomocné pohledy *v_photo*, obsahující kromě dat o fotografii informaci o oblíbenosti a počtu komentářů, pro jednodušší získání dat při vypisování na webu. Pohled *v_tag* obsahuje název značky a počet použití.

5.3.1 MariaDB

Pro persistentní uložení dat je zvolen relační SQL databázový server MariaDB 10.2[5]. Schéma databáze je vytvořeno podle sekce 5.3 za pomocí migrací dostupných v Yii2 frameworku. Každá migrace značí změnu schéma databáze a dovoluje tak i u již běžící aplikace, měnit strukturu databáze. Migrace je reprezentována PHP třídou, kde jsou definovány jednotlivé operace (práce s tabulkami, přidávání záznamů, aj.). Díky migracím je možné udržet databázi verzovanou a při výrazně změně struktury poskytují funkce pro transformaci již existujících dat.

SQL injection Pro všechny dotazy na databázi je využívána třída *ActiveRecord*, které kóduje všechny atributy, čímž znemožní útok SQL injection popsaný v kapitole 5.5.2.

Cache schémy Jelikož si modely získávají seznam atributů dynamicky, dotazem na schéma tabulky při každém vytvoření objektu, je pro zvýšení rychlosti tento výsledek ukládán do dočasné cache paměti aplikace, aby nemuselo docházet k častým dotazům na databázi. Po aplikování migrace se tato paměť musí smazat, aby nedošlo k nekonzistencím.

5.3.2 Elasticsearch

MariaDB nemá v základu podporu pro GPS data a práci s nimi. Z toho důvodu byla použit nástroj Elasticsearch.

Jedná se o databázový nástroj primárně určený pro fulltextové vyhledávání a další složitější dotazy. Díky rozsáhlým slovníkům umožňuje vyhledávat i skloňovaná či špatně napsaná slova. Vyhledávací jádro je postaveno na Apache Lucene, ke kterému přidáva REST API, pro snadnou integraci do systému. Důležitou vlastností pro tuto aplikaci je podpora geolokace a funkce pro výpočet vzdálenost, agregace a získání bodů na základě ohraničujících GPS bodů [6].

Pro práci s daty je potřeba vytvořit model, obsahující funkci pro mapování atributů, vytvoření indexu a obnovu dat. Funkce pro mapování dokumentu místa je zobrazena na výpisu 5.1 a obsahuje celočíselnou identifikace dokumentu a kategorie, do které je přiřazen, pozici definovanou typem *geo_point*, a čas zaznamenání ve standardním databázovém formátu.

5.4 Prototyp

Před grafickým návrhem uživatelského rozhraní došlo k vytvoření wireframu, volně přeloženo jako *drátový model*. Model definuje rozmístění funkčních elementů na stránce. Pokud tedy není předem určeno rozložení stránky, je ideální vytvořit wireframe, protože jeho realizace je značně jednodušší a případné úpravy nejsou tolik náročné, jako ve výsledném grafickém návrhu.

[5] Dostupný na https://mariadb.com

```
public static function mapping() {
  return [
    static::type() => [
      'properties' => [
        'id' => ['type' => 'integer'],
        'location' => ['type' => 'geo_point'],
        'id_category' => ['type' => 'integer'],
        'photo_captured_at' => [
          'type' => 'date',
          "format" => "yyyy-MM-dd HH:mm:ss",
        ],
      ],
    ],
  ];
}
```

Výpis 5.1: Příklad mapování atributů modelu pro Elasticsearch.

Protože cílem aplikace je mít co nejjednoduší uživatelské rozhraní, došlo k vytvoření interaktivního prototypu z wireframu. Prototyp byl otestován uživateli na několika testovacích scénářích, popsaných v sekci 5.4.3.

Interaktivní prototyp byl vytvořen pomocí nástroje Marvell[6]. Aplikace umožňuje vytvářet interaktivní grafické návrhy, na kterých lze otestovat uživatelská přívětivost. Nejprve jsou vytvořeny jednotlivé stránky aplikace a ty následně provázány odkazy, určením aktivní plochy, kde při kliknutí dojde ke změně stránky.

5.4.1 Mapa

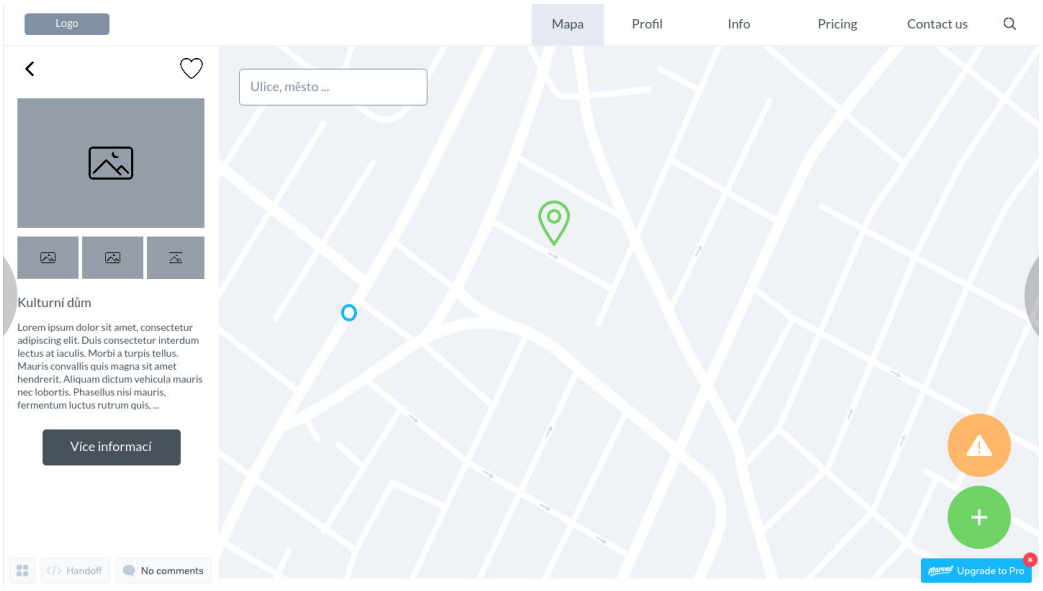
Na stránce je kladen důraz na samotnou mapu, proto zabírá většinu obrazovky. V prvotním návrhu se v levé části se nacházela dynamicky měnitelná část, ve které byl zobrazen detail fotografie při kliknutí na značku, nebo výpis několika prvních fotografií z aktuálního okna mapy. U fotografie byl popis, případně další informace o pořízení a související refotografie. V pravém dolním rohu byla dvě tlačítka, horní singnalizovalo že přihlášený uživatel má nahráné fotky, ale zatím nebyly publikovány. Druhé tlačítko sloužilo k vyvolání okna pro nahrání nových fotografií. Prvotní návrh je zobrazen na obrázku 5.5.

Prvotní návrh počítal pouze s jedním místem na řádku, ale po testování rozhraní došlo k navýšení na dvě místa na řádek. Navíc bylo přidáno tlačítko, které otevírá detail místa v novém okně mapy, protože výsuvný prvek nebyl uživateli přijat i z hlediska optimalizace pro vyhledávače je potřeba stránka s pevnou URL.

5.4.2 Editor

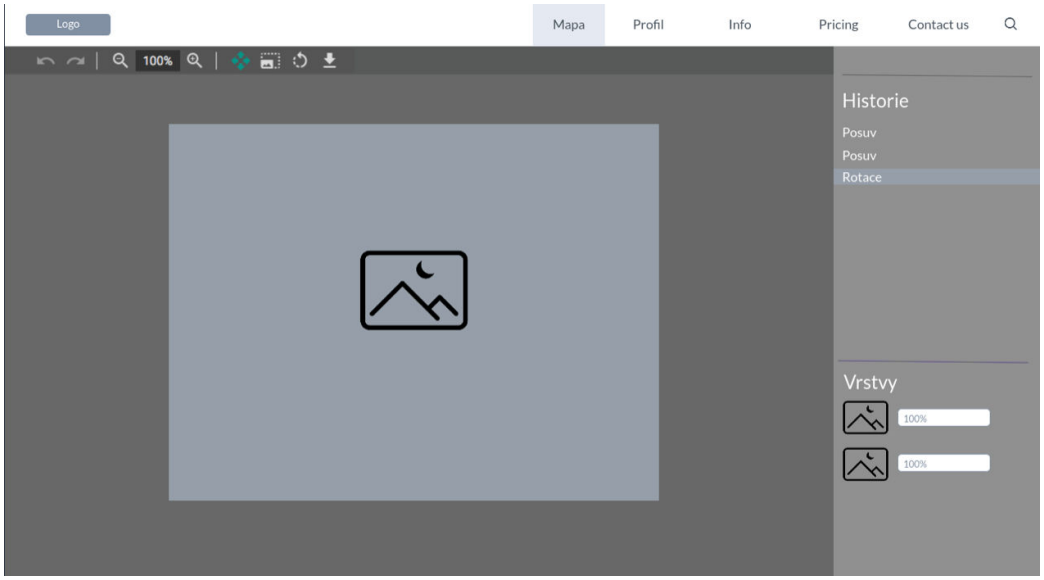
Rozložení editoru je inspirováno desktopovým programem Photoshop [4]. V horní části se nachází lišta s tlačítkem zpět a obnovit pro navizaci v historii, velikost plátna a jako poslední výčet použitelných nástrojů. V pravé části je seznam posledních kroků historie, se zvýrazněným aktuálním stavem. Kliknutím na jiný stav dojde k jeho vyvolání. Posledním

[6]Prototypovací nástroj Marvel: https://marvelapp.com/



Obrázek 5.5: První návrh hlavní stránky s mapou

prvkem je přehled vrstev s náhledem obrázku a polem pro definování viditelnosti. Největší část obrazovky zabírá plátno.



Obrázek 5.6: Wireframe editoru

5.4.3 Testované scénáře

Uživatelům bylo zadáno několik úkolů a k rozhraní systému jim nebylo nic řečeno, kvůli ověření, zda jsou jednotlivé akce intuitivní a tačítka na správných místech. Zpětná vazba získána od uživatelů odhalila nedostatky u rozložení prvků stránky a přinesla návrhy vylepšení pro příjemnější prožitek při používání webu.

Jednotlivé scénáře jsou popsány v následujících sekcích. Obsahují vždy celé znění úkolu předanému uživateli a souhrnné vyhodnocení všech výstupů.

Přihlášení

Úkol: *Poveďte registraci uživatele a následné přihlášení.*
Většina uživatelů hledala poprvé tlačítko pro přihlášení v pravém horním rohu. Protože však lišta s akcemi byla inspirována rozložením podle material designu [3], nacházela se ve spodní části obrazovky.

Spodní lišta byla přesunuta do horního menu a vyhledávač byl vložen přímo do mapy, kde je na první pohled také viditelnější.

Vložení fotografie

Úkol: *Poveďte vložení nové fotografie a zvolte pozici kdekoli v Brně.*
S nalezením výrazného tlačítka pro přidání nové fotografie neměli uživatelé problém. Všichni označili celý poces nahrání za velmi intuitivní.

Zobrazení detailu fotografie

Úkol: *Zobrazte detail libovolné fotky a následné vložení do oblíbených.*
Opět všichni uživatelé zvládli akci bez problémů. Fotky na mapě však byly reprezentovány pouze bodem a byl vznesen návrh, aby mapa obsahovala malé náhledy fotografií, což však z důvodu optimalizace a množství stahovaných dat ze serveru ebylo implementováno.

Úprava v editoru

Úkol: *Vyberte 2 fotografie pro editaci, přejděte do editoru, přesuňte první obrázek a druhému nastavte 50% průhlednost.*
Tlačítku v prototypu nebylo možné přidat nápovědu při přejetí kurzorem, takže uživatelům nebylo jasné zda se jedná o správné tlačítko pro přidání do editoru. Ve výsledném systému bude popis akce zvýrazněn.

U funkcí editoru se uživatelé rychle zorientovali a rozložení hodnotili kladně. Samotné používání funkcí bude nutné otestovat až na funkčním řešení.

5.5 Zabezpečení

5.5.1 Filtrování a validace vstupů

Všechna data, která uživatel zadá pomocí formulářů nebo odešle jiným způsobem do systému pro zpracování jsou vždy považována za potenciálně nebezpečná. Pokud vstupní data nejsou validována, může dojít k útokům, např. *Interpreter Injection*, útok *locale/Unicode* a *buffer overflows*. [1]

Kontrola integrity

Spočívá v ověření, že zadané údaje nebyly podvrhnuty a jsou stejné jako dříve. [1] Při přenosu dat z méně důvěryhodného zdroje je nutná kontrola předaných parametrů. Například u použití pomocné hodnoty ve skrytých polích formulářů nebo přesměrování na aplikaci

třetí strany, kde v požadavku předáváme identifikační údaje. Mohlo by se stát, že útočník v odpovědi pošle jiný identifikátor a pokud by nebyla provedena kontrola vráceného identifikátoru mohlo by dojít k ovlivnění dat, která do tohoto procesu nevstupovala.

Kontrola integrity je v systému řešena na úrovni modelů pomocí scénářů, kdy jsou přesně definovány atributy, jež mohou být naplněny ze vstupních dat. Identifikátory a pevné atributy jsou nastaveny podle aktuální stránky přímo v kódu, ne načtením přijatých dat.

Validace

Kontrola zda jsou data zádána v požadovaném formátu, mají správný typ, délku a obsahují pouze povolené hodnoty. [1] U validace jsou tři možné strategie:

- Přijímat pouze známé hodnoty
- Odmítnout pouze známé chyby
- Přijímat vše *(velmi nebezpečné)*

Pokud systém přijímá pouze známé hodnoty, jsou stanovena pravidla. Když hodnota splní pravidla, je přijata. Druhý typ validace, kdy je definován seznam zakázaných hodnot je nebezpečný. Může se stát, že seznam potencionálně chybných hodnot je nekonečný a není tak možné všechny definovat.

Rychlejší validaci můžeme dosáhnout kombinací obou přístupů. Na začátku proběhne kontrola na malé množství zakázaných pravidel a nemusí docházet ke kontrole všech požadovaných.

5.5.2 Kódování výstupu

V závislosti na kontextu by mělo docházet ke kódování všech dat zadaných uživatelem před zobrazením a nebo přenosem do jiných systémů. Například v kontextu HTML kódovat všechny speciální znaky jako například < a >, aby nedošlo k ovlivnění výsledné stránky.

XSS Cross-site scripting neboli XSS je typ útoku, kdy uživatel donutí stránku spustit vlastní kód. [13] Například místo jména vloží do formuláře javascriptový kód, který projde validací a kontrolou integrity, protože se jedná o řetězec. Při následném vykreslení jména na stránce dojde ke spuštení skriptu. Zakódováním k tomuto spuštění nedojde a kód se pouze vypíše jako posloupnost znaků.

SQL Injection Útok zaměřený na databázový systém. Využívá možnosti zadat místo parametru speciání znaky pro ukončení aktuálního dotazu a připojit za něj další, který může např. smazat veškerá data nebo získat citlivé údaje.

Ochranou před tímto útokem je opět správné kódování vstupních dat od uživatele.

5.5.3 Šifrovaná komunikace

Tato část zabezpečení není obstarána systémem, ale serverem na kterém je spuštěn. Při komunikaci mezi uživatelem a serverem může docházet k výměně citlivých dat. Aby tato data nemohla být odposlechnuta nikým dalším, je potřeba použít šifrovanou komunikaci.

Pro zabezpečenou komunikaci je potřeba použít HTTPS používající TLS protokol s asymetrickou kryptografií pro šifrování. Asymetrická kryptografie používá k šifrování dva

typy klíčů: soukromý a veřejný. To co se zašifruje veřejným klíčem, lze dešifrovat pouze pomocí soukromého a opačně. Soukromý klíč je uložen na serveru a veřejný je předán klientovi ve formě certifikátu při navázání komunikace. Uživatel tak veškerá data zasílaná na server šifruje veřejným klíčem serveru a dešifrovat je může pouze server. [18]

5.6 Multijazyčnost

Aplikace má globální zaměření, necílí na jeden region, a tak jsou všechny texty ve výchozím stavu v angličtině. Pro zpříjemnění práce s aplikací, je implementován systém pro multijazyčnost. Díky němu je možné, přidáním několika souborů, zpřístupnit nový jazyk.

5.6.1 Překlady textů

Všechny texty jsou ve zdrojových souborech před vypsáním předány funkcí *Yii:t()*, které vypíše řetezec v požadovaném jazyce. Překlady jsou uchovávaný v PHP souborech, dělených do kategoriií podle jednotlivých částí aplikace (např. homepage, map, editor). Při načtení tohoto souboru je vytvořeno asociativní pole, kde je klíčem text v angličtině a hodnotou překlad. Každý jazyk má svůj vlastní adresář, pojmenovaný podle kódu jazyka (cs, en), ve kterém se dále nachází jednotlivé soubory kategorií. Pro změnu jazyka aplikace tedy stačí, aby se texty získávaly z adresáře podle kódu aktuálně zvoleného jazyka.

```
return [
  'Explore places' => 'Prozkoumejte místa',
  'Search' => 'Vyhledat',
  'Explore' => 'Prozkoumat',
  'Type city or zipcode' => 'Napište město nebo směrovací číslo',
];
```

Výpis 5.2: Příklad souboru s překlady.

5.6.2 Accept-Language hlavička

Při první návštěvě je jazyk aplikace určen pomocí *Accept-Language* hlavičky zaslané v HTTP požadavku. Ta obsahuje seznam nejvhodnějších jazyků pro daného uživatele. Každý jazyk je označen atributem $q =< 0, 1 >$, značícím váhu (vhodnost) daného jazyka. Hlavička je nastavena automaticky podle, jazyka prohlížeče, zařízení, případně sítě do které je připojen. Používá se tedy pouze při první návštěvě a pokud není jazyk určen jiným způsobem. Na ukázce 5.3 obsahuje hlavička požadovaný jazyk francouštinu a pokud by nebyla dostupná angličtinu, následně němčinu.

```
Accept-Language: fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5
```

Výpis 5.3: Príklad HTTP hlavičky Accept-Language

Pokud již uživatel jednou stránku navštívil, je poslední zvolený jazyk uložen do session a cookie. Při následujícím požadavku bez jazyka, je použit takto uložený jazyk.

5.6.3 Jazyk v URL

Uživatel může zvolit jazyk aplikace pomocí URL na kterou přistupuje. Za určením domény je jazyk jako první atribut. Příklad 5.4 odkazuje na stránku *map* v českém jazyce.

```
https://example.com/cs/map
```

Výpis 5.4: Příklad jazykové URL

Výchozí jazyk (angličtina), má všechny adresy bez jazyka a pokud uživatel přejde na adresu */en* je jazyk nastaven, uložen a uživatel přesměrován na adresu bez jazyka. Pro změnu jazyka tedy stačí aby uživatel otevřel stránku s požadovaným jazykem, ten je následně uložen a použit pro další procházení webu.

Kapitola 6

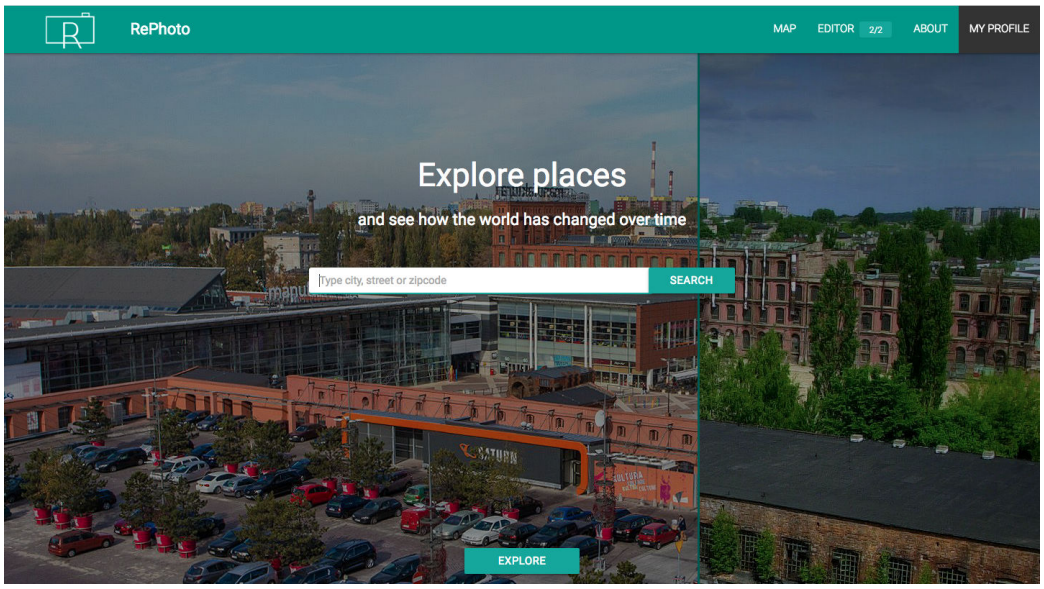
Implementovaná aplikace

V této kapitole je popsáno webové rozhraní určené pro uživatele. Sekce pojednávají o jednotlivých částech webu, rozdělených podle funkčnosti. Některé stránky mohou mít omezenu funkčnost, podle toho, jestli je uživatel přihlášen či nikoli. Vyjímku tvoří sekce SEO (6.6), která popisuje vlastnosi webu podporující větší návštěvnost díky lepšímu propojení s vyhledávači.

6.1 Úvodní stránka

Úvodní stránka webu, musí uživatele upoutat, na první pohled představit službu a umožnit co nejjednodušší interakci pro přechod k hlavní funkčosti webu. Změnit návštěvníka v zákazníka, tedy uživatele, který bude aktivně využívat aplikaci a není to jeho poslední návštěva.

Pozadí úvodní stránky tvoří animace, která kombinuje starší fotografii a novější refotografii. Spolu s nadpisem je tím uživateli prezentováno zaměření aplikace. Ve středu stránky se nachází textový vstup propojený s mapou 6.3, kde po zadání adresy dojde k automatické přepnutí stránky na mapu s požadovanou pozicí.



Obrázek 6.1: Úvodní stránka aplikace

Pod hlavní fotografií je výčet ukázkových míst pro detailnější prezentaci funkcí aplikace. Tyto místa mají vyplněné refotografie a uživatel může vidět co lze s fotografiemi dělat. Ukázky také slouží jako motivace uživateli aby nahrál nové místa případně refotografie.

Dalším prvkem je krátká prezentace funkce mapy, editoru a odkaz pro registraci a přihlášení. Na jedné stránkce je tak jednoduchý popis co může uživatel na webu dělat a případně se prokliknout na funkce, které ho zaujala.

Vzhled úvodní stránky je zobrazen na obrázku 6.1.

6.2 Uživatelský profil

Hlavní část aplikace tvoří místa a fotografie nahrávané uživateli, a proto je potřeba tyto uživatele identifikovat a umožnit jim co nejjednodušší práci se systémem. Přidání míst a fotek může pouze přihlášený uživatel s potvrzeným účtem. Potvrzení účtu probíhá pomocí emailu, obsahující odkaz s náhodně vygenerovaným klíčem ověřující vlastnictví emailové adresy.

Při registraci uživatel zadává pouze nejnutnější informace: celé jméno, emailovou adresu a heslo. Vyplněné údaje je možné změnit na stránce profilu, v sekcí *osobní*. Pro zapomenuté heslo je možné vygenerovat odkaz, který se odešle na přihlačovací email. Po otevření stránky je zobrazen formulář na zadání nového hesla. Heslo lze změnit na jeden odkaz pouze jednou. Po přihlášení je uživateli zpřístupněna věškerá funkčnost systému, viz diagram případů užití (obrázek 5.3).

Registrací je uživateli vytvořen veřejný profil, seskupující všechny nahrané refotografie a obrázky.

6.2.1 Facebook API

Facebook je platforma s mnoha uživateli a je velmi pravděpodobné, že potencionální návštěvnící webu zde již mají založený účet. Protože při registraci do této aplikace zadávali všechny potřebné údaje, mohou být při spárování účtu data přenesena a uživatel je nemusí zadávat znovu. Navíc bude účet použit, prokud by chtěl nějaké místo nebo fotku sdílet na Facebook.

Pro přihlášení se používá Facebook SDK[1]. To umožňuje uživateli přihlásit se pomocí svého účtu aniž by přihlašovací údaje procházeli přes webserver aplikace. Při požadavku na přihlášení je pomocí SDK vygenerována unikátní URL, na kterou je uživatel přesměrován. URL je vytvořena pomocí identifikačních klíčů aplikace spárované se systémem. Uživatel zadá své přihlšovací údaje a odešle je na servery Facebooku. Pokud je uživatel autorizován, je do systému navrácen přístupový token. Na straně aplikace dojde pomocí SDK k ověření, zda je token platný a jsou získány informace o uživateli.

Spolu s osobními informacemi (celé jméno, email) je získáno také *facebook_id*, jednoznačně identifikující uživatele. Při přihlášení se zjišťuje, zda už neexistuje uživatel s touto identifikací a pokud ano je použit. Další scénář, který může nastat je, že se uživatel nejprve zaregistruje na webu a poté přihlásí přes Facebook. V tu chvíli nemá vyplněno *facebook_id*, ale pokud se emaily shodují dojde ke spárování účtů. Pro přihlášení přes facebook musí mít uživatel email ověřený, je tedy zaručeno, že uživatel je vlastníkem emailové adresy.

[1] Facebook SDK https://developers.facebook.com/docs/reference/php/

6.3 Mapa

Mapové podklady jsou použity z Google Maps[2], kvůli podrobné mapě a dostupné API, která dovoluje upravit mapu vloženou do aplikace. Styl je při první návštěvě ve výchozím stavu, kdy se zobrazuje grafická mapa. Tlačítkem však jde přepnout na satelitní zobrazení.

6.3.1 Vyhledávání

Díky knihovně *Places* a třídám *SearchBox*, *Autocomplete* je do mapy přidáno vyhledávací pole s automatickým doplněním ulic, měst a míst. Při psaní do políčka, dochází k odesílání API dotazu na místa obsahující zadaný řetězec, výsledky jsou dobrazeny v seznamu pod ním. Po výběru místa dojde k posunutí okna mapy a získání aktuálních míst.

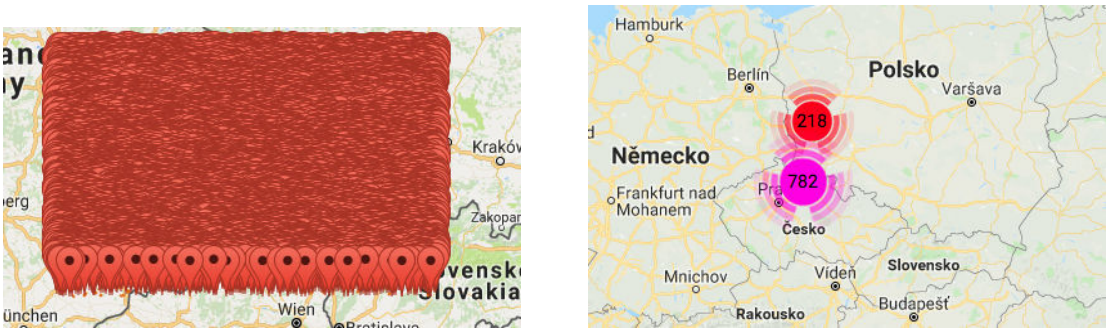
6.3.2 Místa

Jednotlivá místa jsou na mapě reprezentována bodem, kterému při vkládání do mapy stačí určit souřadnice. Každému bodu je navíc přiřazen vlastní atribut *id*, identifikující místo v databázi. Pokliknutí na bod, dojde k odeslání požadavku s *id* na server a ten vrátí náhled místa. Náhled obsahuje nejstarší fotku, nejnovější fotku, data pořízení, název a tlačítko pro přechod na samostatnou stránku místa.

Při otevření stránky s mapou, dojde k získání polohy všech míst v aktuálním okně mapy, ale pouze 6 náhledů. Je to z důvodu rychlejšího přenosu. Body na mapě požadují pouze pozici, odpovídající dvěma číslům a identifikaci. Náhled však obsahuje název a hlavně 2 fotografie, které se stahují na samostatný požadavek. Pokud by tedy na mapě bylo mnoho míst, uživatel by musel stáhnout všechny obrázky, ikdyž je nemá v plánu vidět.

6.3.3 Shlukování

Každý obrázek bodu zabírá určité místo a pokud by docházelo k vykreslování stále všech bodů v okne, hrozí přeplnění znázorněné obrázkem 6.2. GoogleMap API implementuje pro tento případ knihovnu *MarkerClusterer*, vytvářející shluky blízkých bodů, aby nedošlo k překryvu. Všechny body jsou nejprve vykresleny na mapu a poté nejbližší nahrádí obrázku shluků s počtem reprezentovaných bodů znázorněným číslem uprostřed.



(a) bez shlukování

(b) se shlukováním

Obrázek 6.2: Zobrzaení míst na mapě

[2]Dostupné na https://www.maps.google.com

Při implementaci *MarkerClusterer*-u jsem však narazil na limity při velkém množství bodů. Shluky se počítají až na straně klienta, po přidání všech bodů na mapu. Problémem jsou místa s velkým počtem bodů, případně výraznější oddálení mapy, kdy dojde k vykreslení všech míst na planetě.

Řešením je vytvářet shluky na serveru. Data jsou však uložena v relační databázi, která neobsahuje funkce pro práci s GPS souřadnicemi a ruční počítání by bylo výpočetně a implementačně náročné. Pomocí Elasticsearch byl vytvořen mezistupeň, který je pomocí komplexního dotazu (Výpis 6.1) schopen agregovat blízká místa.

Součástí serverového dotazu je pozice levého horního a pravého dolního rohu okna, používáná pro získání míst v aktuální okně mapy. Filtr používá funkci *geo_bounding_box* pro kontrolu, zda místo leží v žádané oblasti.Vyfiltrovaná místa jsou rozdělena do shluků pomocí *Geohash grid_aggregation*. Každému místu je vypočítán GeoHash [7] a globálně je určeta přesnost dotazu, které spojí blízká místa. Přepočet přesnosti na plochu shluku je znázorněn v tabulce 6.1. Poslední fází je získání středu oblasti pomocí funkce *geo_centroid*.

```
{
  "aggregations": {
    "zoom-in": {
      "filter": {
        "geo_bounding_box": {
          "location": {
            "top_left": [51.1223880, 21.7539468],
            "bottom_right": [47.1298424, 14.1309432]
          }
        }
      },
      "aggregations": {
        "zoom1": {
          "geohash_grid": {
            "field": "location",
            "precision": 3
          },
          "aggregations" : {
            "centroid" : {
              "geo_centroid" : { "field" : "location" }
            }
          }
        }
      }
    }
  }
}
```

Výpis 6.1: Elasticsearch dotaz pro získání míst ohraničených z určité oblasti a spojených do shkluků s přesností 3.

Přesnost	Plocha shluku	Rozsah přiblížení Google mapy
1	5,009.4km x 4,992.6km	<1,4>
2	1,252.3km x 624.1km	<5,6>
3	156.5km x 156km	<7,8>
4	39.1km x 19.5km	<9,10>
5	4.9km x 4.9km	<11,12>
6	1.2km x 609.4m	<13,14>
7	152.9m x 152.4m	<15,16>
8	38.2m x 19m	<17,18>
9	4.8m x 4.8m	<19,20>
10	1.2m x 59.5cm	<21, 22>
11	14.9cm x 14.9cm	-
12	3.7cm x 1.9cm	-

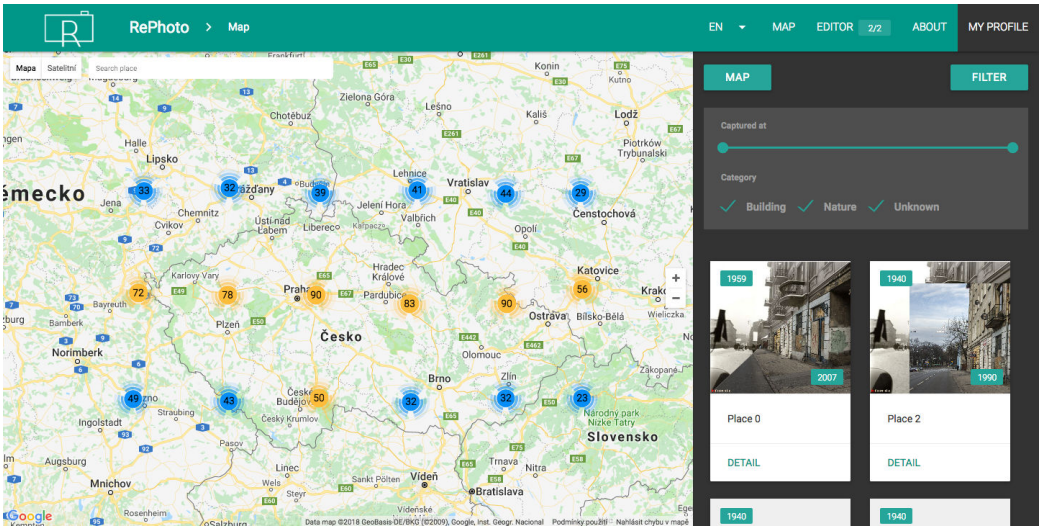
Tabulka 6.1: Přepočet přesnosti Geohash agregace

6.3.4 Filtr

Pro specifičtější vyhledávání míst je vytvořen filtr obsahující kategorii a rok pořízení fotografie. Místa uložená do Elasticsearch databáze, obsahují mimo polohy, také pole dat vytvoření všech fotografií a identifikaci kategorie. Kategorie jsou pevně dány: budova, příroda, neurčeno. Dotaz při vyhledávání obsahuje filtr kategorie, reprezenované polem identifikátorů, protože každá kategorie může být zapnuta individuálně. Rozsah roku pořízení používá noUiSlider[3] a místo je zobrazeno, pokud alespoň jedna fotografie vyhovuje rozsahu.

Součástí filtru je tlačíko *Mapa*, které umožňuje zobrazit (resp. skrýt) mapu, čímž nedochází k filrování podle okna mapy, ale jsou zobrazena všechna místa v systému. Implementačně se jedná o skrytý *checkbox*, kterému tlačítko mění hodnotu.

Výsledná stránka s mapou je zachycena na obrázku 6.3.



Obrázek 6.3: Výsledný vzhled stránky s mapou

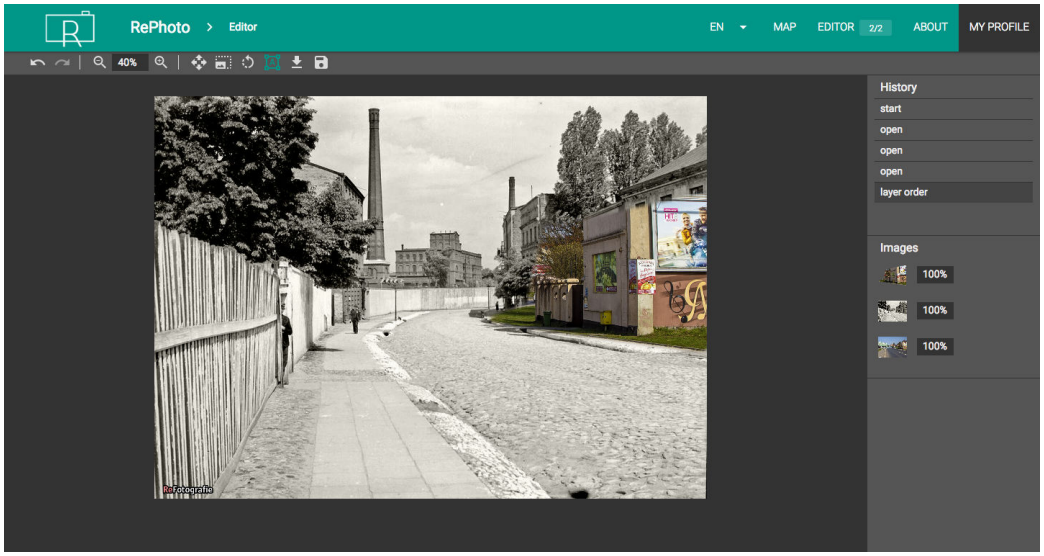
[3]Dostupný na https://refreshless.com/nouislider/

6.4 Editor

Základ editoru tvoří HTML element *cavas*, do kterého jsou vykresleny požadované fotografie. Následné transformace jsou řízeny pomocí Javascriptového objektu *Canvas*, vytvořeného pro práci s editorem. Všechny funkce editoru jsou zpracovávány pomocí Javascriptu přímo v prohlížeči uživatele. Operace tak nejsou zpožděny komunikací se serverem.

Transformace popsané v sekci 3.1 jsou aplikovány při každém vykreslení obrázku. Vždy se pracuje se stejným zdrojem, objektem *Image*[4], který je součástí objektu *CanvasImage* obsahující metody pro manipulaci v editoru a hodnoty pro výpočet tranformačních funkcí.

Výsledná stránka editoru je zachycena na obrázku 6.4. Horní lištu tvoří tlačítka jednotlivých funkcí a pravou stranu přehled všech změn, společně s výpisem vrstev obrázků v editoru. Editor zabírá vždy celou plochu okna prohlížeče. Pokud je obrázek větší než dostupná plocha, jsou u obsahové části zobrazeny posuvníky, aby došlo k zachování celistvosti stránky.



Obrázek 6.4: Výsledný vzhled stránky editoru

6.4.1 Inicializace

Inicializace editoru je velmi jednoduchá. Stačí předat blokový element do konstruktoru třídy *Canvas*. Při tvorbě objektu dojde k vygenerování celkového rozložení editoru, včetně tlačítek a funkcí pro zachytávání událostí. Není nutné manuálně vytvářet prvky editoru, knihovna se o vše postará automaticky. Vytvoření vlastní knihovny umožňuje také aktualizovat editor, beze změny prvků stránky, protože potřebné elementy jsou vygenerovány automaticky.

```
var editor = new Canvas(document.getElementById('editor'));
canvas.addObject(new CanvasImage('https://example.com/img.png'));
```

Výpis 6.2: Inicializace editoru na blokovém prvku s ID editor.

[4]Dokumentace https://developer.mozilla.org/en-US/docs/Web/API/HTMLImageElement/Image

6.4.2 Třídy Canvas a CanvasImage

Veškerou funkčnost editoru zapouzdřuje třída *Canvas*. Obsahuje všechny potřebné atributy jako velikost plátna, seznam obrázků, jednotlivé elementy tlačítek funkcí a pomocné proměnné pro určení aktivního prvku.

Nad plátnem jsou odchytávány události s kurzorem. Při kliknutí dojde k detekci kolize, kdy je na každém vykresleném a viditelném obrázku na plátně zavolána funkce *contains*. Této funkci se předá aktuální pozice kurzoru a návratová hodnota *true/false*, určuje zda na této pozici je obrázek vykreslen. Kolize se kontrolují na seznamu obrázků v opačném pořadí, protože ty jsou vykresleny později, tedy výše.

Pozice kurzoru na plátně je vypočítána z aktuální pozice v okně, ke které je připočtena hodnota posunutí plátna a odečtena velikost odsazení plátna od kraje prohlížeče.

Jednotlivé obrázky jsou reprezentovány třídou *CanvasImage*. Obsahuje atributy pro transformační funkce, tedy hodnoty aplikované rotace, pozici na plátně a samostatnou vrstvu s obrázkem. Pokud je nějaká vrstva vybrána pro aplikování funkce (platná kolize s kurzorem), nastaví se tomuto obrázku značka *active* na *true*, a podle aktivní funkce editoru je volána transformační metoda přímo na tomto objektu.

6.4.3 Základní funkce

Historie

Po vykonání transformace nebo posunu je do seznamu *history* v objectu *Canvas* zaznamenán aktuální stav objektu. Při navracení v historii se aktuální stav nahradí stavem z historie a vynutí se překreslení plátna. Objekt obsahuje navíc atribut *historyPointer*, který uchovává pozici aktuálně vykresleného stavu. Při návratu v historii tak nedojde k okamžitému smazání nového stavu. Každý nový stav je přidán za aktuální pozici indexu a všechny další jsou odstraněny.

Pořadí vrstev

Objekt *Canvas* si uchovává seznam obrázků na plátně v atributu *objects*, kde pořadí hraje vliv na vykreslení. Čím vyšší index obrázek má, tím později je na plátno vykreslen. Při přesunu obrázku o vrstvu výše dojde k seřezení seznamu a překreslení plátna.

Uživatel má díky *jQueryUI* a *widgetu Sortable*[5] možnost měnit pořadí vrstev. Každý element fotografie je možné označit kliknutím kurzoru a přesunutím na požadované místo. Na obalujícím elementu je zachytávána událost *DOMChanged*, která značí že došlo ke změně vnořeného obsahu. Při této události dojde k získání všech elementů o jednu úroveň níže. Dříve vložené obrázky se nacházejí na konci seznamu, a proto se průchod provádí v opačném pořadí. Původní seznam se pouze přeskládá, protože v přehledu vrstev nejsou uložen celé objekty, každý element má však atribut *dataindex*, který značí pořadí v seznamu obrázků. Po změně pořadí je vrstvám tento atribut změněn.

Průhlednost fotografie

Třída *Image*, které se předává URL obrázku, nemá vestavěnou funkci na úpravu průhlednosti. Pro změně hodnoty viditelnosti fotografie je nutné získat zdrojová data, kde první tři hodnoty odpovídají intenzitě dané barvy (červená, zelená, modrá) a každá 4. hodnota

[5] Dostupný na https://jqueryui.com/sortable/

reprezentuje alfa kanál, tedy hodnotu od 0 do 1. Po úpravě se data předají zpět do objektu, hodnota se uloží do objektu *CanvasImage* a dojde k překreslení plátna.

Posun obrázku

Při kliknutí do plátna se vypočítá aktuální pozice kurzoru a zjistí se, zda je v tomto místě vykreslen nějaký obrázek. Pokud ano a je zvolen nástroj *přemístění*, dojde k vybrání objektu obrázku, podle první zjištěné kolize. Při každém posunutí kurzoru se změní hodnoty *x,y*, značící souřadnice levého horního rohu obrázku. Nové hodnoty jsou vypočítány jako stará hodnota sečtena s přírustkem předchozí pozice kurzoru s aktuální pozicí.

Rotace

Podobně jako u posunu obrázku, akorát s tím rozdílem, že se neupravují hodnoty *x,y*, ale atribut *rotation* značící úhel natočení.

Změna velikosti

Pokud je zvolen nástroj *změna velikosti*, dojde k vykreslení pomocné mřížky kolem fotografie s 8 body pro změnu velikosti. Body v rohu obrázku transformují velikost podle obou os a body ve středu stěn pouze podle jedné. Levá horní strana mění atributy *x,y*, tedy počáteční pozici vykresení obrázku, a pravá dolní strana atributy *width, height*, značící šířku a výšku obrázku. Tato transformace umožňuje deformovat obrázek a změnit poměr stran.

Výřez oblasti

Hlavní funkcí editoru je kombinování fotek. Pro označení jaké oblasti budou u fotografií zvýrazněny je implementována funkce výběru pomocí tvorby polygonu. Při každém kliknutí kurzoru je do seznamu uložena aktuální pozice bodu a po dokončení cesty je vytvořena nová vrstva obsahující pouze vybranou oblast.

První vložený bod zobrazí kružnici s poloměrem 5px. Tato oblast slouží pro ulehčení ukončení výběru, aby uživatel nemusel kliknout na přesný začátek. Při každém dalším kliknutí do plátna dojde k vykreslení čáry od posledního bodu do aktuální pozice.

Po uzavření cesty je automaticky vytvořena nová instance *CanvasImage* a vložena na jako nejvyšší vrstva. Získání vybrané části z plátna probíhá ve 4 krocích.

Krok 1 Nejprve se vypočítá minimální ohraničující obdelník vybrané oblasti, tzn. průchod všemi body a získání minimální a maximální pozice *x,y*. Od minimálních hodnot je odečteno 20px a k maximálním přečteno 20px, aby výsledný obrázek měl odsazení.

Krok 2 Dále je vytvořeno nového plátno, na které se vykreslí zadaná cesta, vyplněná černou barvou. Na toto plátno je aplikováno rozmazaní, díky kterému dojde ke zjemnění hran a výsledný obrázek bude mít plynulejší přechod. Kvůli tomuto byly přidany 20px okraje.

Krok 3 Následně je hlavní plátno nastaveno do režimu *destination-in*. [5] Nastavením této kompozitní operace dojde k tomu, že při dalším vykreslení, zůstane na plátně pouze oblast, která je na novém plátně reprezentována černou. Menší intenzita černé znamená průhlednější obraz.

Krok 4 Posledním krokem je získání vykresleného obrázku podle minimálního ohraničení a vytvoření nového *CanvasItem* objektu s odkazem na výsledek. Hlavní plátno je poté překresleno a s novou vrstvou je možné pracovat stejně jako s původními obrázky

Uložit obrázek

Nepřihlášení uživaté mají možnost uložit výsledný obrázek do zařízení. Při kliknutí na tlačítko a potvrzení dialogu dojde k získání dat obrázku pomocí funkce:

- *canvas.toDataURL("image/png")*

Následně je vytvořen dynamický element s odkazem a vyvolána událost kliknutí na tento odkaz. Díky tomu dojde k zobrazení dialogu pro stažení výsledného obrázku.

Příhlášený uživatel může navíc uložit obrázek do profilu a zobrazit jej na stránce místa v sekci upravené obrázky. Na rozdíl od refotografií se tyto obrázky ukládají do zvláštní tabulky *photo_save*.

Přiblížení plátna

Pro detailnější úpravu a přesnější manipulaci s obrázem je možné plátno přiblížit/oddálit. Implementačně se jedná o nastavení procentuální změny velikosti oproti počáteční hodnotě. Změnou dojde k úpravě stylů, přesněji výšky a šířky plátna. Následně k překreslení obsahu, avšak předtím se plátnu nastaví atribut *scale*, táke v procentech, aby vykreslovaný obraz byl zvětšený, resp. zmenšen podle velikosti plátna a nevznikala bílá místa.

6.4.4 Výstup

Pro demostraci funkce editoru je na obrázku 6.6 zobrazen výstup po aplikování funkce výřezu oblasti na historické fotografii 6.5a a nové vrstvy jsou zobrazeny nad novější refotografií 6.5b.

(a) Historická fotografie

(b) Nová refotografie

Obrázek 6.5: Fotografie zpracované v editoru

Obrázek 6.6: Obrázek kombinující fotografie 6.5a a 6.5b

6.5 Přidání refotografie

Aplikace obsahuje předem nahrané historické fotografie, ke kterým mohou uživatelé přidat nové refotografie. Proces přidání je rozdělen do 4 kroků, popsaných v následujících kapitolách.

6.5.1 Krok 1: Nahrání

Uživatel nejprve vybere místo a klikne na tlačítko *Přidat refotografii*. Tím se spustí 4 krokový proces, kde nahraje nový soubor ze zařízení, nebo vybere jednu z dříve nahraných fotografií, u kterých proces nedokončil.

Po zvolení je fotografie uložena do databáze s příznakem *aligned = false*. Ten značí, že fotografie zatím nebyla zarovnána a není zobrazena veřejně na stránce místa.

6.5.2 Krok 2: Zarovnání

Při prvním příchodu na krok 2 je spuštěn algoritmus na automatické zarovnání a uživatel přesměrován na krok 3. Systém chce uživateli co nejvíce ulehčit nahrávání refotografií, proto jsou fotografie automaticky zarovnány. Pokud byla nová refotografie focena z velmi podobné pozice a algoritmus fotky správně zarovná, není potřeba manuální zásah uživatele, který může nejnáročnější operaci přeskočit.

Automatické

Skript pro automatické zarovnání je psán v jazyce Python za pomocí knihovny OpenGL[6] a implementuje algoritmy popsané v sekci 3.2. O jeho volání se stará přímo PHP kód, který skript spustí s třemi parametry a jehož výstupem je hodnota *true/false*, značící zda zarovnání proběhlo bez chyby. První dva předávané argumenty jsou absolutní cesty k původní a nové fotografii a třetí je cesta k dočasnému souboru, kam skript uloží zarovnaný obrázek.

[6]Dostupné na https://www.opengl.org

Cesta je poté transformována na veřejnou URL a vrácena do webového prohlížeče, kde je původní fotografie nahrazena novým zdrojem.

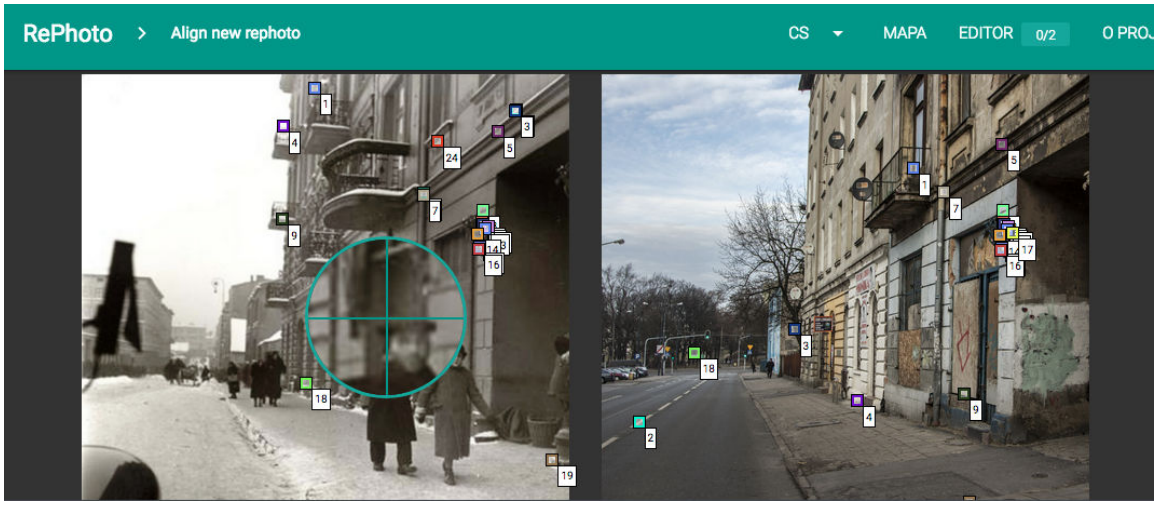
Implementace algoritmu Nejprve jsou pomocí funkce *cv2.imgread()* získána data obou obrázků. Pomocí FAST detektoru (*cv2.FastFeatureDetector_create().detect()*) dojde k získájí klíčových bodů obou fotografií, ze kterých se vypočítají deskriptory zavoláním funkce *cv2.ORB_create().compute()*. Deskriptory jsou porovnány pomocí Flannova algoritmu, implementovaného v OpenCV jako *cv2.FlannBasedMatcher().knnMatch()* a shodné body použity pro vytvoření tranformační matice homografie, díky *cv2.findHomography()*. Ta je aplikována na nově vkládaný obrázek funkcí *cv2.warpPerspective()*.

Poloautomatické

Automatické zarovnání nefunguje ve všech případech a někdy je tedy nutné aby uživatel zadal stejné body fotografií ručně. Při tomto zarovnání jsou fotografie zobrazeny vedle sebe a klikáním střídavě na jednu a druhou dojde k přidávání společných bodů. Jednotlivé body jsou odlišeny barvou a také pořadovým číslem.

Pro přesnější určení místa je k dispozici lupa 6.7. Zobrazí se při přidržení na již vytvořeném bodu. Pokud exituje bod i na druhém obrázku dojde k zobrazení lupy i zde, uživatel tak přesně vidí na jaké místo původní bod umístil.

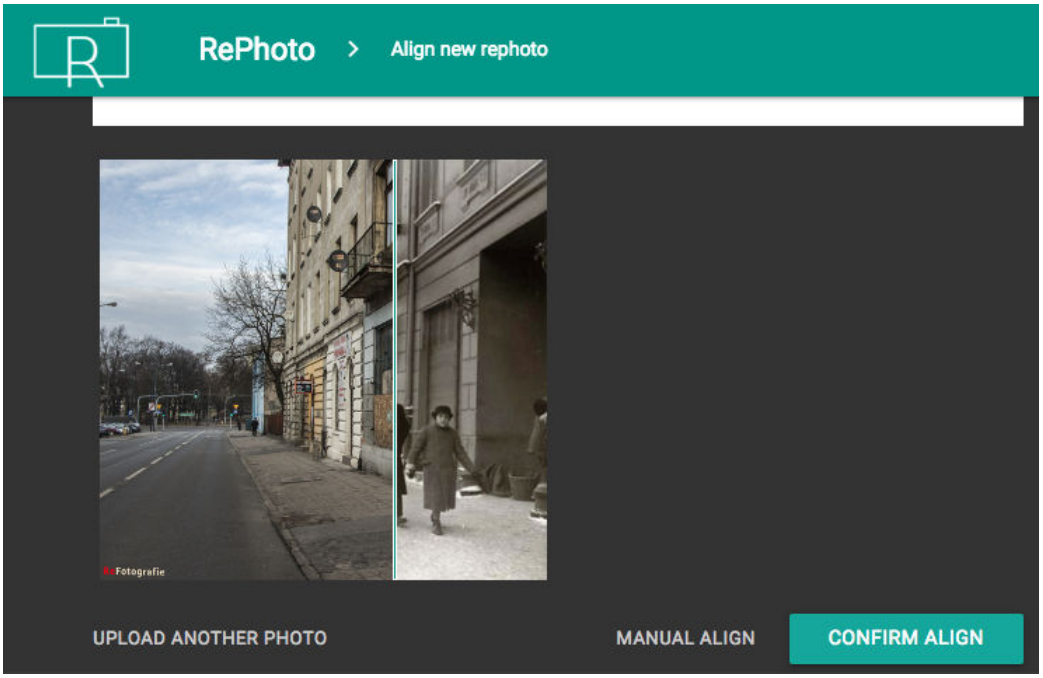
Klinutím na tlačítko *Zarovnat*, dojde k odeslání zvolených bodů na server, kde je spuštěn Python skript pracující s OpenCV knihovnou implementující stejný algoritmus jako automatické zarovnání (3.2) s rozdílem, že se nevyhledávají body automaticky, ale jsou použity ručně zvolené.



Obrázek 6.7: Rozložení při manuálním zarovnání.

6.5.3 Krok 3: Kontrola

Předposledním krokem přidání nové fotografie je kontrola zarovnání. Na stránce je zobrazena starší fotografie překrytá novou a pohybem kurzoru přes fotografii dochází k zobrazování staré. Uživatel tak díky překryvu jasně vidí, zda je zarovnání správné a pokud ano může postoupit na poslední krok. V opačném případě se vrátí o krok zpět a přidá jiné nebo další body a proces opakuje.



Obrázek 6.8: Kontrola výsledného zarovnání snímku.

6.5.4 Krok 4: Dokončení

Poslední část nahrání tvoří formulář s diplňujícími informacemi o fotce. Některé údaje jako poloha a datum vytvoření je získáno z metadat fotografie (sekce 2.4). Další informace jako název, popisek je zadán ručně. Automaticky doplněná data může uživatel změnit a po vyplnění dojde k uložení refotografie do systému a zobrazení na detailu místa.

Nově nahrané fotografii je v databázi nastavena hodnota *aligned = 1*, avšak *verified = 0* značí že žádný administrátor, prozatím fotografii neověřil. Tato značka je zavedena z důvodu kontroly obsahu, aby uživatelé nenahrávali špatně zarovnané fotky nebo dokonce fotky nesouvisející s tímto místem.

6.6 SEO

Search Engine Optimalization (SEO) neboli optimalizace pro vyhledávače je v dnešní době velmi důležitá vlastnost webu, díky které se může zlepšit viditelnost stránek ve výsledcích vyhledávání. Cílem je získat co nejvíce neplacených návštěvníků na web.

Hlavním faktorem je časté používání klíčových slov ve vlastním textu stránky, avšak stránky tohoto webu jsou ve většině případů automaticky generované detaily místa a je potřeba se zaměřit na statické stránky: úvodní stránku, mapu a profil.

Na úvodní stránce je hlavním prvkem animace fotografií na pozadí, kterého si všimne uživatel. Pro vyhledávač jsou důležitá klíčová slova: *refotografie*, *místa* a *časem změnila* (*rephotography*, *places* a *change over time*). Další sekcí na stránce je stručný odstavec popisující podstatu stránky tak, aby obsahoval věty které uživatelé mohou chtít vyhledávat.

6.6.1 Robots.txt

Soubor *robots.txt* dostupný v hlavním adresáři webu (*https://example.com/robots.txt*), je, jak už název napovídá, určen pro roboty procházející stránku za účelem analýzy klíčových slov a zavedení stránek do výsledků vyhledávání. Obsah souboru definuje, které stránky mají roboti ignorovat, například odkazy na dynamické akce, které nemají výstupní stránku, ale pouze mění stav aplikace nebo stránky administrace, které nejsou veřejně přístupné.

V případě tohoto webu je zákázáno procházet a indexovat stránky jejichž URL začíná */admin*. Obsahuje také odkaz na mapu webu popsanou v následující kapitole 6.6.2.

6.6.2 Sitemap.xml

Stejně jako robots.txt je shromažďuje soubor sitemap.xml informace o webu. Obsahuje odkazy na všechny stránky, které má robot indexovat a usnadňuje tím robotovi procházení. Navíc je zde informace o datu poslední změny, prioritě stránky a jak často by měl robot procházet stránku aby měl aktuální informace.

Tato aplikace obsahuje 6 statických stránek: úvodní, mapa, editor, přihlášení, registrace, obnova hesla a dále pak dynamicky tvořené stránky míst s prioritou 0.8.

Na výpisu 6.3 je příklad detailu místa, který byl naposledy upraven 18.4.2018, četnost úpravy je v rámci měsíce a priorita je nastavena na 0.8 z 1.

```
<url>
        <loc>http://rephoto.local/cs/place/view?id=2</loc>
        <lastmod>2018-04-18T18:53:39+00:00</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.8</priority>
</url>
```

Výpis 6.3: Příklad stránky v souboru sitemap.xml

6.6.3 Meta značky

Jedná se o speciální značky HTML jazyka, využívané zpracovávané většinou roboty pro lepší prezentaci webu na stránkách třetích stran. Základními SEO značkami jsou titulek, popis doplněné o sociální značky Open Graph[7].

Titulek stránky se určuje značkou *title* a ikdyž se nejedná přímo o meta značku, tvoří velmi důležitou součást SEO. Společně se značkou *<meta content=d̈escription>̈* definují náhled webu ve výsledcích vyhledávání, tzv. Search engine result page (SERP).

Nastavení těchto značek je velmi důležité pro vyhledávání a pozici stránek ve vyhledávačích, jsou hodnoty vyplňovány pouze v administraci. Při tvorbě místa uživatel vyplní název a popis, avšak administrátor může hodnotu značek změnit nezávisle na uživatelově nastavení.

6.7 Administrace

Součástí aplikace je také administrátorské rozhraní, umožňující jednoduchý přehled všech registrovaných úživatelů, míst a fotografií v tabulkách. Administrace je dostupná na ad-

[7] Open Graph http://ogp.me

rese */admin*. Přístup je umožněn pouze uživatelům s rolí administrátor (sloupec *role = 1*, uživatel je *role = 0*).

Administrace a uživatelská část mají oddělené přihlašování. Sezení (*session*) na serveru je vytvořena pro každou aplikaci pod vlastním klíčem, např. administrace má *_identity-admin*.

Administace je dělena do 4 sekcí: dashboard, uživatelé, místa a fotografie. Po úspěšné autorizaci je zobrazena úvodní obrazovka - dashboard. Ostatní sekce jsou dostupné z menu v hlavičce.

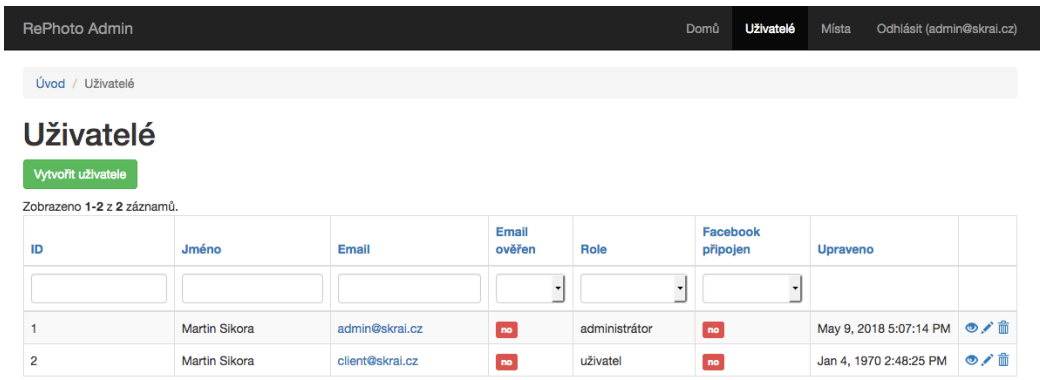
6.7.1 Dashboard

Dashboard je vstupní stránka administrace. Slouží k vizuálnímu zobrazení nejdůležitějších údajů o systému, případně úkolů co je potřeba splnit. Všechny prvky by měly být zobrazeny na jedné stránce v přehledných blocích.

Administrace nebude dostupná pro širokou veřejnost, ale pouze pro pár správců, protože obsahuje pouze doplňkové funkce. Dashboard proto seskupuje počet nahraných fotografií a míst, počet registrovaných uživatelů a přehled posledně nahraných fotografií a míst čekajících na schválení.

6.7.2 Zobrazení a editace dat

Uživatelé, místa a fotografie mohou být prohlíženy a editovány z administrace. Každá sekce má skupinu stránek (CRUD) s akcemi. Hlavní je vždy přehled všech dat v tabulce. Používá se třída *GridView*, zobrazující data vyfiltrovaná data pomocí *ActiveDataProvider*. Ten v základu obsahuje pomocné funkce pro stránkování, kdy je při tvorbě objektu přidám pouze objekt databázového dotazu a poté dojde k načtení dat podle požadované stránky. Do datazázového dotazu jsou automaticky přidány části *limit* a *offset*, značící počet získávaných prvků, respektivě pořadí prvního s odsazením z vyhovujících výsledků.



Obrázek 6.9: Stránka s přehledem uživatelů.

Pod hlavičkou tabulky se nachází filtrační formulář. Typ dat určuje, jaký vstupní prvek je pro filter použit:

- Textový vstup
- Rozbalovací nabídka

- Výběr data

u každého řádku tabulky jsou dostupná tlačítka pro přechod na detail prvku, editaci a smazání. Před smazáním musí uživatel potvrdit vystakovací okno aby nedošlo k nechtěnému smazání.

6.8 API

API tvoří samostatnou aplikaci s vlastním nastavením a zcela odlišným stylem komunikace. Používá REST rozhraní a komunikace probíhá pomocí HTTP metod *GET*, *POST*, *PUT* a *DELETE*. Veškerá data jsou přenášena ve standardním formátu *JSON*. Tato aplikace je nejčastěji využívána pro propojení s mobilní aplikací.

Aplikace je dustupná v adresáři */api*. Pro umožnění jednoduchých změn ve struktuře a zároveň zachování zpětné kompatibility, nejsou zrojové kódy přímo v adresáři aplikace, ale jsou děleny do modulů podle čísla verze. První verze je dostupná na adrese s prefixem *https://example.com/api/v1*. Mobilní aplikace jsou vyvíjeny s nejnovější verzí API a server zajišťuje, že po vydání nové verze nepřestanou aplikace fungovat, pouze nebudou mít dostupně nové funkce.

Jednotlivé sekce jsou rozděleny do datových celků a až na vyjímku sekce 6.8.1 popisují koncové body a strukturu komunikace.

6.8.1 REST

Representational State Transfer (REST) je styl architektury pro výměnu dat mezi distribuovanými systémy. Používá se pro snadný přístup ke zdrojům, což jsou v tomto případě data a stavy aplikace, které jsou definovány pomocí vlastní URL. REST implementuje čtyři základní operace se zdroji a to vytvoření (create), získání (read), úpravu (update), smazání (delete), spojované pod zkratkou *CRUD*. [19]

Získání

Pro získání zdrojů se používá metoda *GET* a případné filtrování je pomocí atributů v URL. Nejčastější formát vrácených dat je ve formátu *json*. Příklad URL adresy pro získání seznamu míst je:

- GET /api/v1/places

Adresa neobsahuje identifikaci konkrétního místa a tak dojde k vrácení všech, avšak na adrese:

- GET /api/v1/places/1

Data jsou dostupná pro detail konkrétního místa s identifikací 1.

Tvorba Jelikož konkrétní zdroj není ještě vytvořen, je definován koncový bod, na který jsou pomocí metody *POST* odeslána data. Například pro vytvoření místa je adresa definována stejně jako pro získání seznamu:

- POST /api/v1/places

V těle požadavku je navíc zaslán json objekt s daty pro uložení.

Úprava Tato akce se provádí pomocí metody *PUT*. Jediný rozdíl mezi tvorbout a úpravou je URL koncového bodu. Při úpravě má zdroj již přiřazenou identifikaci a ta je tedy reprezentována v URL, např.:

- PUT /api/v1/places/1

Data jsou, stejně jako u tvorby, zasílána ve formátu *json*.

Smazání Poslední používanou metodou je *DELETE*, fungující stejně jako *GET*. Přistupuje se pouze na URL mazaného zdroje:

- DELETE /api/v1/places/1

6.8.2 Struktura odpovědi

Protože požadavek na API odpovídá protokolu HTTP, je stavový kód odpovědi prvním znakem, zda proběhlo vše v porádku. Systém používá kódy:

- **200** Vše v porádku
- **400** Chyba zaslaných dat
- **401** Chybná autorizace
- **403** Nepovolený přístup
- **500** Chyba serveru

Pokud se vyskytla chyba, je v těle odpovědi objekt s atributem *status* nastaveným na *false* a chybová zpráva dostupná pod atributem *error*. Při úspěšném zpracování požadavku je *status = true*. Pokud jsou součástí odpovědi data, vyskytují se pod atributem *data* s typem *seznam* nebo *objekt*.

6.8.3 Autorizace

Ikdyž se jedná o veřejné API, je pro některé funkce vyžadováno přihlášení. Autorizovat se mohou všichni uživatelé stejnými údaji jako na webu. Pokud ověření údajů proběhne v pořádku je navrácen přístupový kód, který je potřeba posílat v hlavičce požadavku pod označením *Authorization*. Jako typ autorizace se používá *Bearer* s vlastním tokenem vygenerovaným serverem. Typ *Basic* není použit z důvodu, že přihlašovací jméno a heslo je posíláno při každém požadavku a ikdyž je komunikace šifrována použitím HTTPS protokolu, je to více zranitelnější než náhodně vygenerovaný token, který s přihlašovacími údaji nemá nic společného. Ty se posílají pouze při prvním požadavku.

Přihlášení je realizováno dotazem: *POST /api/v1/user/login*

6.8.4 Dokumentace

Protože je API otevřené rozhraní pro komunikaci jiných aplikací se systémem, tvoří důležitou část dokumentace. Ta popisuje všechny dostupné koncové body aplikace. U každého bodu je definována požadovaná metoda se strukturou dat dotazu a u odpovědi výčet stavových kódu. Atributy objektů mají přesně definovány datové typy a také přiznak, zda se jedná o povinný parametr či nikoli.

```
{
  "email": "admin@skrai.cz",
  "password": "KLMnf8678B"
}
```

Výpis 6.4: Data požadavku přihlášení.

```
{
  "success": true,
  "data": {
    "access-token": "iwAwRR..",
    "valid_to": 1526495100
  }
}
```

Výpis 6.5: Data odpovědi přihlášení.

Dokumentace byla tvořena v jazyce RAML[8]. Zdrojový kód je možné transformovat do klikatelné webové stránky nebo strukturovaného dokumentu. Umožňuje také definovat objekty, které je možné znovupoužívat, čímž se zjednoduší a zrychlí tvorba dokumentace. Případné změny objektů se také promítnou na všech místech dokumetu.

Kompletní dokumentace je kvůli rozsahu dostupná pouze na přiloženém CD ve formátech:

- HTML - */src/api/modules/v1/docs/api.html*
- PDF - */api-dokumentace.pdf*

Ukázka HTML souboru je zobrazena v příloze C.

[8] Dostupný na https://raml.org/

Kapitola 7

Nasazení

Tato kapitola popisuje požadavky na server a jednotlivé kroky pro spuštění aplikace (7.1). Je zde také popsáno virtualizované spuštění pomocí Docker (7.2).

7.1 Server

Pro běh aplikace je potřeba webový server *Apache*. Tento server je použit, protože aplikace obsahuje soubor *.htaccess* s úpravami směrování a povolenými moduly. Konkrétně se jedná o přesměrování, pokud požadována adresa nezačínající doménou třetího řádu *www.* a dále slouží pro určení, jaká aplikace bude zpracovávat požadavek. Dále je potřeba nainstalovat PHP verze 7.1 nebo vyšší. PHP 5.6 není podporováno kvůli použití nových konstrukcí jazyka. Zpracovávání pomocí PHP 7.1 má také velmi nižší pamětovou náročnost a celkové spracování požadavku je rychlejší.

Pro shlukování míst a rychlejší vyhledávání na mapě je potřeba nainstalovaný Elasticsearch 5.3.2. Po instalaci je nutné změnit adresu a přihlašovací údaje pro komunikaci. Toto nastavení probíhá v souboru *common/config/main.php*. Pokud je komunikace ustanovena, je možné spustit jeden příkaz 7.1, díky kterému dojde k vytvoření indexů a naplnění dat z relační databáze.

```
php yii elasticsearch/init
```

Výpis 7.1: Příkaz pro spuštění skripnu pro inicializaci Elasticsearch databáze, vytvoření indexů a nahrání dat.

Před samostaným spuštěním aplikace je potřeba nainstalovat všechny požadované knihovny. Toto uleží nástroj *composer*[1]. Spuštěním příkazu *composer install* dojde k získání všech potřebných knihoven a jejich návazností. Knihovny, které mají být staženy jsou vyjmenovány v soubouru *composer.json*, který je dostupný v hlavním adresáři aplikace.

Důležitým krokem je také inicializace Yii2 aplikace pomocí spuštění skriptu *php init*, nacházející se také v hlavním aresáři webu. Tento skript vygeneruje všechny konfigurační soubory a nastaví přístupová práva podle požadovaných norem.

Posledním krokem, před samotným používáním, je vytvoření databázových tabulek a nahrání počátečních dat. Pro tento krok je vytvořena akce v konzolové aplikaci, spuštěná příkazem *php yii migrate/up*, která spouští migrační soubory seřazené podle názvu.

Po nainstalování těchto aplikací a spuštěním skriptů je aplikace připravena k použití.

[1] Composer https://getcomposer.org

7.2 Docker

Pro testovací účely je vytvořen konfigurační soubor pro spuštění virtuálního prostředí pomocí nástroje *Docker*[2]. Umožňuje vytvářet kontejnery obsahující požadované prostředí pro běh aplikací aniž by tyto služby byly dostupné v hostujícím systému. Tím je umožněno jednoduché a rychlé nasazení na různé architektury, stačí pouze nainstalovaný Docker. Komunikace s kontejnerem probíhá stejně jako s jiným zařízením v síti.

7.3 CI/CD

Zrojové kódy jsou verzovány pomocí nástroje git. Ten přináší výhodu při tvorbě v týmu, avšak v tomto případě slouží prozatím spíše jako záloha. Používá se repozitář na platformě GitLab[3], obsahující i další pomocné funkce, ne jen uložiště pro zdrojové kódy, např. postupnou integraci a nasazení (Continuous inregration and deployment).

CI/CD podporuje vývoj, usnadňuje zakomponování nových funkcí do existujícího projektu a přidává možnost automatického publikování nových verzí webu. Pro tento projekt jsou použity nástroje GitLab-u. V hlavním adresáři aplikace se nachází konfigurační soubor, ve kterém jsou definovány procesy spuštěné po každém spojení vývojové větve.

7.3.1 Verzování

Proces vývoje je přesně definován a při dodržování umožňuje automatizovat celou řadu věcí. Po počátečním nastavení serveru stačí pouze vyvíjet nové funkce a zavedené změny jsou automaticky testovány a nasazovány na požadované servery.

Hlavní větev repozotáře *master* odpovídá verzi, která je aktuálně nasazená na produkčním prostředí. Do této větve se změny nepřidávají přímo, ale zařazuje se pouze vývojová větev určená k produkčnímu nasazení. Každé zavedení je označeno značkou verze zvané *tag*. Druhou větví je vývojová *develop*, ta obsahuje všechny ukončené, schválené a otestované větve s novou funkčností a opravami chyb. Při každém zavedení do této větve jsou spušteny automatizované testy a pokud nenastane chyba, jsou změny zavedeny, jinak je potřebná oprava a znovuzavedení.

Pokud by byl projekt zveřejněn jako opensource je větev master označena jako *chráněná*, značící že pro zavedení změn je potřeba vytvořit *požadavek* a pouze vybrané osoby mohou tuto změnu začlenit. Stejně bude označena i vývojová větev, čímž se zajistí regulace nových funkcí a oprav pro konkrétní verzi aplikace.

7.3.2 Průběh vývoje

Proces je rozdělen do 3 kroků:

Vývojová větev Každé zavedení do vývojové větve kontaktuje server, na kterém běží *GitLab runner*, nasadí aplikaci a následně spustí testy.

Produkční větev Po zavedení změn do produkční větve dojde opět ke kontaktování serveru a web je nasazen na dočasné dočasné prostředí, kde dojde k manuálnímu otestování,

[2]Docker https://www.docker.com
[3]GitLab https://gitlab.com

zda je opravdu vše v pořádku. Zdrojové kódy nejsou plně pokryty testy a tak je tento krok nutný.

Produkční prostředí Zavedení změn na produkční prostředí je možné z administrace GitLab-u díky propojení s runnerem na serveru, konfigurovaném v souboru *gitlab.yml*. Narozdíl od předchozích kroků se tento neděje automaticky a je spuštěn pouze po manuálním otestování a kliknutím na tlačítko *Nasadit*.

Tímto je proces nasazení výrazně zjednodušen. Není potřeba se přímo připojovat na server a nahrávat na něj změny ručně. Veškerá konfigurace se nachází v jednom souboru a jedniným potřebným krokem je nainstalovat na začátku projektu *runner* a propojit ho s konkrétním repozitářem.

Kapitola 8

Testování

Tato kapitola popisuje tvorbu automatizovaných testů a případy užití manuálních testů, které byly prováděny pro kontrolu správné funkčnosti aplikace.

8.1 Automatické testování

Pro automatické testování byly vytvořeny jednotkové a funkční testy. Tyto testy pomáhají při vývoji, kdy se jednotlivé části aplikace přepisují a přidává se funkčnost. Jednotkové testy se zaměřují vždy na určitou funkci a kontrolují systém na nejnižší úrovni, zda každá funkce zpracovává data správně a nedochází k nežádoucím výjimkám.

Cílem je pokrýt automatizovanými testy kažkou funkci a část aplikace, avšak u této aplikace jsem se ze začátku zaměřil na testování nejdůležitých části. Tento systém nebyl vyvíjen testy řízeným vývojem a tak byly testy vytvářeny zpětně, hlavně kvůli častým změnám v architektuře.

Zdrojové kódy testů se dále dělí v každé aplikaci na jednotkové (unit) 8.1.2, funkční (functional) 8.1.3, akceptační (aceptance) 8.1.4.

8.1.1 Codeception

Pro automatické testování byl použit framework *Codeception*[1]. Každá aplikace (frontend, admin, console, api) má vlastní rozhraní pro testy. Zdrojové kódy jsou umístěny vždy v adresáři *app/tests*. Jednotlivé testy jsou tak oddělené a spouští se samostatně. Pro zjednodušení je vytvořen v hlavním adresáři aplikace soubor *codeception.yml*, který obsahuje pouze odkazy na aplikační testy a složí pouze jako spojení. Samostatně neobsahuje konfiguraci testovacího prostředí a pouze spojuje konfigurace všech aplikací. Spuštěním testů s tímto konfiguračním souborem dojde k ověření všech aplikací a tento přístup je použitý kvůli CI/CD 7.3. Jedná se o vemi výrazné usnadnění při ověření funkčnosti aplikace po přídání funkčnost a automatické nasazení nové verze.

8.1.2 Jednotkové testy

Každá třída používaná v aplikaci má svoji testovací třídu se stejným názvem a příponou *Test* (např. *Place* a *PlaceTest*). Metody v těchto třídách testují jednotlivé funkčnosti předáním několika různých vstupních dat a porovnávají výstup s očekávaným. Základem testování není pouze vytvořit testy, které budou splněny, ale testy, ověřující co největší škálu vstupů.

[1] Codeception https://codeception.com

Uživatel totiž může odeslat jiný než požadovaný formát dat a aplikace musí umět správně zareagovat a neskončit chybou.

8.1.3 Funkční testy

Funkční testy kontrolují větší celky a u této aplikace jsou využity na testování API 6.8. Nekontrolují pouze jednu funkci, ale jejich propojení (např. proces tvorby místa). Testy jsou děleny podle řadičů (controller), ve kterých jsou testovány akce. Kontroluje se také funkčnost modelů a jejich propojení s databází, zda při uložení modelu na úrovni aplikace dojde k uložení dat do perzistetní databáze a následná kontrola při inicializaci modelu.

8.1.4 Akceptační testy

Nejmenší pokrytí aplikace mají akceptační testy, jelikož se uživatelské rozhraní kvůli vylepšení použitelnosti často měnilo. Akceptační testy budou vytvořeny po vydání první verze aplikace. Ze začátku bude nahrazeno manuálním testování, protože tyto testy jsou zaměřeny na používání aplikace, tak jak ji vidí uživatel. Podstatou testů je otevření webové stránky v prohlížeči a interakce s jejími prvky specifikované pomocí CSS nebo XPath selektorů. Kontroluje se zda navigační prvky fungují správně, při kliknutí na tlačíka dochází k požadovanému chování a po dokončení události jsou zobrazeny informační zprávy.

8.2 Manuální testování

Manuálně je testováno konečné použití aplikace. Hlavním účelem tohoto testování, nepočítámli kontrolu před vytvořením automatizovaných akceptačních testů, je zjištění správného propojení uživatelského rozhraní s funkcemi.

Před implementací byl testován prototyp aplikace a uživatelská zpětná vazba zapracována. Jednalo se však o prototyp. Důležitou funkci má i grafická stránka, která by měla na uživatele působit dobrý dojmem, aby chtěl stránku pravidelně nebo častěji využívat. Důležité je, aby akční tlačítka byla viditelná a rozhraní by mělo uživatele vést plynule k dalším krokům, aby nad aplikací nemusel hodně přemýšlet a jednotlivé prvky jasně definovaly co provedou.

Testování probíhalo zadáním určitého úkolu uživateli a sledováním kde uživatel požadované prvky hledá, jestli po stránce nebloudí a zda se co nejrychleji dostane ke splnění. Pro získání lepší zpětné vazby a možnost zpětně analyzovat jednotlivé akce uživatele, je do stránky implementována funkce *Smartlook*[2], která umožňuje nahrávat pohyb kurzoru po stránce a po ukončení návštěvy vytvoří video pro pozdější analýzu a zjišťování tzv. *heat map*, tedy oblastí kde se uživatel nejčastěji pohybuje kurzorem. Na těchto místech by měla být umístěna nejdůležitější tlačítka a zprávy.

8.2.1 Scénáře

Tato sekce obsahuje výpis jednotlivých testovaných scénářů, které byly předány šesti uživatelům ke splnění. Nadpis značí úkol a pod ním je popsána zpětná vazba získaná od uživatelů společně s popisem chování při plnění úkolů.

[2]Smartlook https://www.smartlook.com

Vytvořte nový uživatelský účet.

Všichni uživatelé zvládli registrovat nový profil, ale jeden měl ze začátku problém najít odkaz pro registraci. Nejedná se totiž o klasické tlačítko, ale pouze text zobrazený na konci přihlašovacího formuláře. Protože pro většinu uživatelů byl prvek jednoduše dostupný, není potřeba zvýraznění.

Přihlaste se do systému.

Tento krok nedělal žádnému uživateli problém, protože přihlašovací okno viděli již v předcházejícím úkolu. Na stránku přešli odkazem v hlavičce, vyplnili údaje formuláře a odeslali formulář.

Odhlašte se.

Tlačítko pro odhlášení se nachází v menu po zobrazení stránky profilu. Uživatelé čekali vysouvací nabídku při přejetí kurzorem nad odkazem *Můj profil*.

Vyhledejte nějaké místo v Brně a zobrazte jeho detail.

Většina uživatelů použila k přechodu na mapu vyhledávač na úvodní stránce, pouze 2 udělali o krok navíc, když otevřeli stránku mapy přes hlavičku a místo poté vyhledali pomocí formuláře v mapě. Pro zobrazení detailu místa použili jak rychlý náhled po kliknutí na bod mapy, tak i náhled v bočním panelu. Přechod přes boční panel je jednodušší, protože obsahuje tlačítko *Detail*. Uživatelé ocenili rychlý náhled při kliknutí na bod mapy.

Zobrazte na mapě pouze místa s fotografiemi z roku 1970 - 2000.

Zobrazení a aplikace filtru mapy byla všemi splněna bez problémů.

Nahrajte novou fotografii pro nějaké místo v Brně.

Nahrávání nové refotografii je nejsožitější akce v systému. Uživatelé však ocenili navigaci v podobě zobrazených kroků s vyznačením aktivního a automatické zarovnání. Při poloautomatickém zarovnání však chybělo přesnější navádění, co v daný moment může uživatel udělat. Nastala situace, kdy chtěl uživatel nejprve zadat body do historické fotografie a potom odpovídající do nové, ale systém vložil pouze jeden bod a poté očekával vložení korespondujícího. Možným vylepšením je překrýt fotografii při vložení bodu průhlednou vrstvou, která fotografii začerní a připadně při kliknutí dojde k zobrazení hlášky, že je potřeba přidat bod ve druhé fotografii.

Kapitola 9

Závěr

V této diplomové práci byla navržena a implementována webová aplikace pro správu refotografií. Analýzou existujících řešení se zjistilo, že podobná aplikace pro refotografie již existuje, ale má jiné zaměření. Na základě stávajících řešení došlo k vytvoření seznamu požadovaných funkcí, které by měl systém obsahovat. Prioritou aplikace je jednoduché uživatelské rozhraní a tak došlo k uživatelskému testování prototypu i grafického návrhu pro získání zpětné vazby a usnadnění používání.

Systém obsahuje funkce uživatelského profilu pro registraci, přihlášení pomocí emailu nebo Facebook učtu, nahrávání fotek, jejich přehled na mapě a uložení mezi oblíbené. Editor umožňuje posouvat, otáčet, měnit velikost a průhlednost fotografií, vytvářet nové vrstvy a výsledek uložit do souboru.

Celý proces od prohlížení míst po nahrání nové refotografie je implementován, systém může být nasazen na produkční prostedí a zveřejněn.

Do dalších verzí aplikace je možné přidávat nové funkce editoru, např. přenesení barevného schéma z historické fotografie na novou, případně deformace pomocí mřížky bodů pro plně manuální zarovnání fotografií. Důležitou částí při spuštění bude také získání počáteční datové sady, která motivuje uživatele k přidání nových refotografií. Nemusí se jednat o místa s existující refotografií, ale pouze historické fotografie a formou hry nalákat uživatele k pořízení a nahrání nových refotografií.

Literatura

[1] Data Validation.
URL https://www.owasp.org/index.php/Data_Validation

[2] The Definitive Guide to Yii 2.0.
URL http://www.yiiframework.com/doc-2.0/guide-index.html

[3] Material Design.
URL https://material.io/guidelines/

[4] Základy práce s pracovní plochou.
URL https://helpx.adobe.com/cz/photoshop/using/workspace-basics.html

[5] CanvasRenderingContext2D.globalCompositeOperation. 2018.
URL https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/globalCompositeOperation

[6] Elasticsearch Reference. 2018.
URL https://www.elastic.co/guide/en/elasticsearch/reference/6.2/geo-queries.html

[7] Elasticsearch Reference. 2018.
URL https://www.elastic.co/guide/en/elasticsearch/guide/current/geohashes.html

[8] Boyer, D. E.; Turner, R. M.; Webb, R. H.: *Repeat photography*. Washington, DC: Island Press, c2010, ISBN 9781597267137.

[9] Brown, A.: Digital Preservation Guidance Note 4, 2008.
URL
https://www.nationalarchives.gov.uk/documents/graphic-file-formats.pdf

[10] Böhmer, M.: *Návrhové vzory v PHP*. Brno: Computer Press, první vydání, 2012, ISBN 978-80-251-3338-5.

[11] Curtin., D. P.: *The textbook of digital photography*. Marblehead, Mass: ShortCourses.com, druhé vydání, 2007, ISBN 19-288-7375-8.

[12] Gentle, J. E.: *Matrix algebra*. London: Springer, springer science & business media vydání, c2007, ISBN 03-877-0872-3.

[13] Grossman, J.: *XSS attacks*. Burlington, Mass.: Syngress, c2007, ISBN 978-1-59749-154-9.

[14] Krig, S.: *Computer Vision Metrics*. Apress, 2014, ISBN 978-1-4302-5929-9.

[15] Lundquist, S.: Image file formats.
URL https://99designs.com/blog/tips/image-file-types/

[16] Mokrejš, J.: Vlnková komprese obrazu. 2008.

[17] Randers-Pehrson, G.: Extensions to the PNG 1.2 Specification, Version 1.5.0.
URL
http://ftp-osl.osuosl.org/pub/libpng/documents/pngext-1.5.0.html#C.eXIf

[18] Ristic, I.: *Bulletproof SSL and TLS*. London: Feisty Duck, 2015, ISBN 978-1-907117-04-6.

[19] Rouse, M.: REST (REpresentational State Transfer). 2017.
URL https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer

[20] Rumbaugh, J.; Jacobson, I.; Booch, G.: *The unified modeling language reference manual*. Boston: Addison-Wesley, druhé vydání, c2005, ISBN 03-212-4562-8.

[21] Sayood, K.: *Introduction to data compression*. San Francisco: Morgan Kaufmann Publishers, druhé vydání, c2000, ISBN 15-586-0558-4.

[22] Sikora, M.: Informační systém pro jazykovou školu. 2015.

[23] Szeliski, R.: *Computer Vision*. London: Springer, 2010, ISBN 978-1-84882-935-0.

Příloha A

Obsah CD

- src/ zdojové soubory
- dp.pdf text diplomové práce
- dokumentace-api.pdf dokumentace k API
- plakat.png plakát popisující aplikaci
- README návod ke spuštění systému

Příloha B

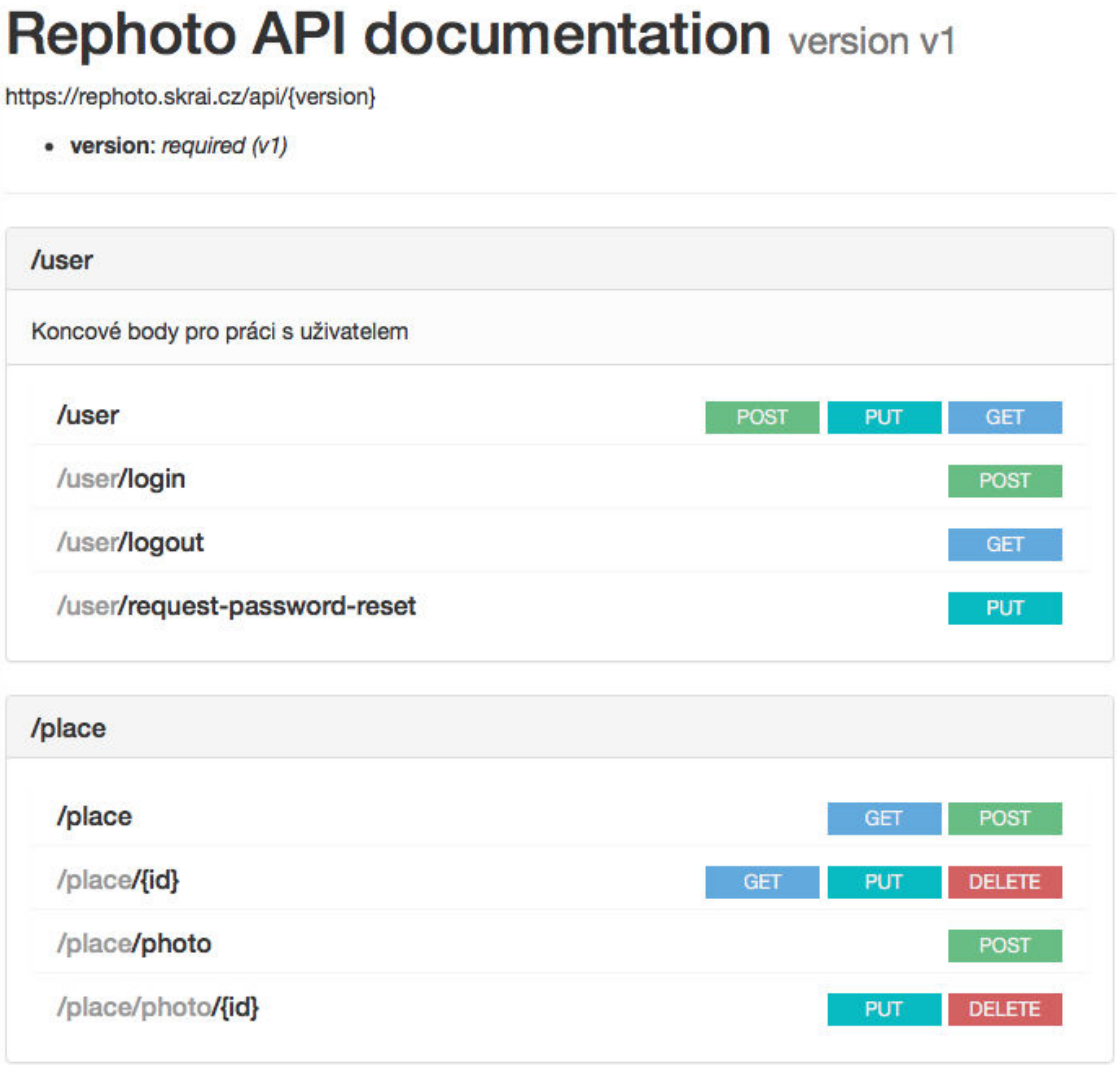
Adresářová struktura aplikace

- api - *API aplikace*
  - components - *komponenty rozšiřující funkčnost aplikace*
  - config - *konfigurační soubory*
  - models - *databázové modely*
  - modules - *moduly verzí*
    - v1 - *API verze v1*
      - controllers - *řadiče*
      - models - *databázové modely*
      - tests - *zrojové kódy testů*
      - codeception.yml - *konfigurační soubor pro spuštění testů*
  - runtime - *dočasné soubory generované při používání aplikace*
  - web
    - index.php - *hlavní skript pro spuštění*
  - codeception.yml - *konfigurační soubor pro spuštění testů všech verzí*
- backend - *administrace*
  - assets - *balíky stylů a skriptů*
  - config - *konfigurační soubory*
  - controllers - *řadiče*
  - messages - *překladové soubory*
  - models - *databázové modely*
  - runtime - *dočasné soubory generované při používání aplikace*
  - tests - *zdrojové kódy testů*
  - views - *pohledy*
  - web - *veřejné soubory*
  - codeception.yml - *konfigurační soubor pro spuštění testů*
- common - *společná část pro aplikace*
  - components - *komponenty rozšiřující funkčnost aplikace*

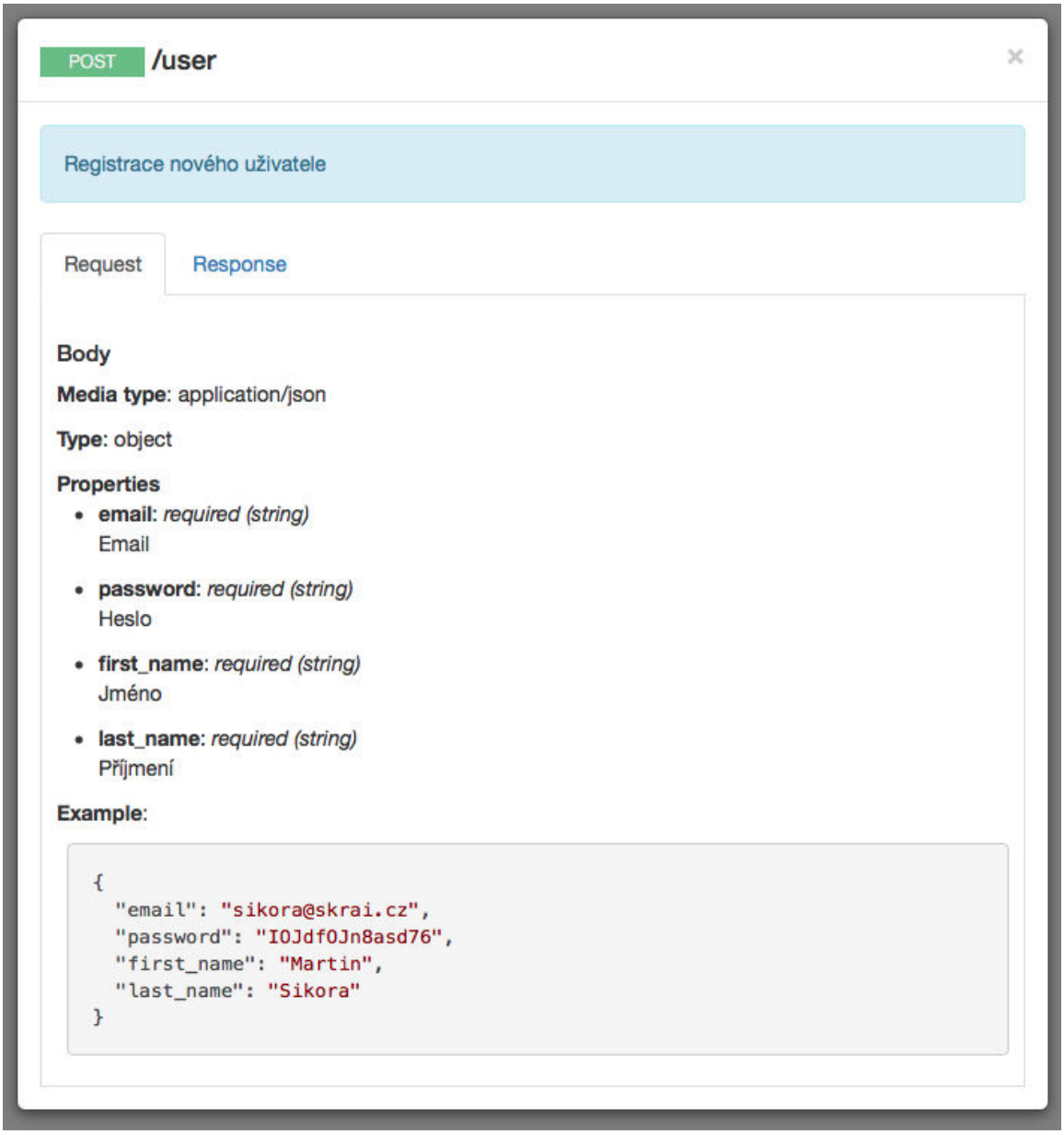
  - config - *konfigurační soubory*
  - mail - *rozložení emailů*
  - models - *databázové modely*
  - tests - *zdrojové kódy testů*
  - widgets - *balíky funkcí*
  - codeception.yml - *konfigurační soubor pro spuštění testů*
- console - *konzolová aplikace*
  - config - *konfigurační soubory*
  - controllers - *řadiče*
  - migrations - *databázové migrace*
  - models - *databázové modely*
  - runtime - *dočasné soubory generované při používání aplikace*
- frontend - *uživatelská část*
  - assets - *balíky stylů a skriptů*
  - components - *komponenty rozšiřující funkčnost aplikace*
  - config - *konfigurační soubory*
  - controllers - *řadiče*
  - messages - *překladové soubory*
  - models - *databázové modely*
  - runtime - *dočasné soubory generované při používání aplikace*
  - tests - *zdrojové kódy testů*
  - views - *pohledy*
  - web - *veřejné soubory*
  - widgets - *balíky funkcí*
  - codeception.yml - *konfigurační soubor pro spuštění testů*
- opencv - *skripty pro práci s OpenCV knihovnou*
  - align.py - *automatické zarovnání fotografie podle zdrojové*
  - align-points.py - *zísání bodů pro automatické zarovnání*
  - align-custom.py - *automatické zarovnání s vlastními body*
- uploads - *uživateli nahrané obrázky*
- vendor - *knihovny nainstalované pomocí nástroje composer*
- .htaccess - *konfigurační soubor webového serveru*
- codeception.yml - *konfigurační soubor pro spuštění všech testů*
- composer.json - *požadované knihovny*
- init - *inicializační skript aplikace*
- yii - *skript pro spuštění konzolové aplikace*

Příloha C

Ukázka stránky s API dokumentací



Obrázek C.1: Přehled koncových uzlů a dostupných metod API.



Obrázek C.2: Ukázka stuktury požadavku v API dokumentaci.

Příloha D

Plakát

Obrázek D.1: Plakát pro prezentaci.